



Integration of Bluetooth-enabled Sensors into E-Health Application

for Home Healthcare and
Monitoring.

AGUSTÍN GARCÍA PÉREZ

*MASTER THESIS 30 ECTS, ERASMUS 2007-08 SWEDEN
THESIS N° 1/2008, BIOMEDICAL ENGINEERING*

Integration of Bluetooth-enabled Sensors into E-Health Applications for Home Healthcare and-Monitoring.

Agustín García Pérez

Master Thesis

Subject Category: Technology

Series Number Communication and Programming 2007-08

University College of Borås

School of Engineering

SE-501 90 BORÅS

Telephone +46 33 435 4640

Examiner: Fernando Seoane Martínez

Supervisor: Fernando Seoane Martínez

Client: Högskolan i Borås

Date: 2008 March 28th

Keyword: Blood Pressure, Body Weight, Home Healthcare,
Portable, Bluetooth.

ABSTRACT

Information and telecommunication technologies have been used for medical applications for almost a century: the radio, the phone and the television have been used to provide medical assistance to remote areas like boats located in the sea, oil rigs and other isolated locations. This new use of ICT was given the name of Telemedicine.

Since the 90's, health care systems have been looking for new ways and alternatives to provide health care than the traditional hospital centers. Information technologies and communication systems have been developed, especially with the deployment of Internet access to most of residential areas, and have brought health care to the electronic age; Carving a new term E-Health.

Nowadays Telemedicine is used in several E-health applications for teleconsultation, telediagnosis, telemonitoring and has been successfully tested in teleradiology, telecardiology, teledermatology, telepsychiatry, etc. This technology is used in rural areas, health areas, prisons, home cares, emergencies, wars, etc.

This project implements two different parts. The first one is an user application to measure weight and blood pressure from the patient using two Bluetooth-enable measurement devices: the UC-321PBT Precision Health Scale and UA-767PBT Blood Pressure Monitor. The second one implements a web server to store all patients reports with updated information in real time. This information can be consulted through a web page for medical staff in HL7 patient's report format.

PREFACE

This final degree work has been done in Borås, Sweden, during the course 2007/2008 under the European Exchange Program Socrates in execution of the Erasmus agreement between Högskolan I Borås and Universidad Politécnica de Madrid, and more specifically the agreement between the Institutionen Ingenjörshögskolan and the Escuela Universitaria de Ingeniería Técnica de Telecomunicación.

The project work has been supervised by Dr. Fernando Seoane Martinez and with the collaboration with the Department of Telematic Engineering at the Polytechnic University of Madrid, Spain.

CONTENTS

ABSTRACT	III
PREFACE	V
CONTENTS.....	- 1 -
ACKNOWLEDGEMENT.....	- 3 -
1. INTRODUCTION	- 5 -
1.1 RENAL FAILURE AND HOME-MONITORING	- 6 -
1.2 BLOOD PRESSURE	- 7 -
1.3 BODY WEIGHT	- 8 -
1.4 E-HEALTH SYSTEM OVERVIEW	- 9 -
2. DATABASE AND INFORMATION ACCESS	- 11 -
2.1 SYSTEM DATABASE	- 11 -
2.1.1 Patient Local Database.....	- 13 -
2.1.2 Server Database.....	- 14 -
2.2 SYSTEM OVERVIEW AND INFORMATION ACCESS.....	- 15 -
3. SYSTEM SECURITY	- 19 -
3.1 EDF FILES	- 19 -
3.2 DATA ENCRYPTION STANDARD (DES)	- 20 -
3.3 SECURE SHELL (SSH).....	- 22 -
4. ELECTRONIC PATIENT RECORD	- 25 -
5. MANAGEMENT SOFTWARE.....	- 27 -
5.1 COMMUNICATION AND MEASUREMENT SUITE.	- 27 -
5.2 WIZARD SUITE.....	- 29 -
5.3 SYSTEM ARCHITECTURE	- 30 -
5.4 PATIENT'S SIDE	- 31 -
5.4.1 User Interface	- 32 -
5.4.2 User Application	- 36 -
5.5 SERVER APPLICATION	- 44 -
6. REFERENCES.....	- 59 -
7. APPENDIX.....	- 61 -
APPENDIX I. UC-321PBT PRECISION HEALTH SCALE SPECIFICATIONS	- 61 -
APPENDIX II. UA-767PBT BLOOD PRESSURE MONITOR SPECIFICATIONS	- 62 -

ACKNOWLEDGEMENT

Ante todo quería agradecer a mis padres, Teresa y Agustín, por todo el apoyo que me han brindado durante este tiempo y por las facilidades que me han dado para poder sacarme este título de ingeniero técnico que tanto he deseado.

A mi hermana Teresa, por estar siempre ahí cuando la he necesitado y empujarme y animarme siempre a seguir adelante.

A Ana, por todo su apoyo durante estos tres últimos años, tanto en los buenos como en los malos momentos.

A toda mi familia, que siempre ha estado ahí preocupándose y preguntándome por como iba con la carrera.

A los que también considero como familia, todos mis compañeros y amigos, con los que he compartido momentos increíbles tanto dentro como fuera de la universidad.

Y por supuesto a Fernando, una estupenda persona tanto como profesor como de amigo y que me ha dado la posibilidad de llevar a cabo este proyecto de la mejor manera posible, compartiendo conmigo todo su conocimiento y experiencia.

Por todo ellos y a todos vosotros. **¡¡¡¡ MUCHAS GRACIAS !!!!**

Agus

Borås (Sweden) April 1st, 2008

1. INTRODUCTION

This project is focused on the field of medicine, more specifically to telemedicine. The main aim of this work is to develop an E-Health application which allows patients, who need a continued monitoring of weight and blood pressure, to stay at home. This way they skip regular visits to the health center or hospital to take measurements and take the measurements at home.

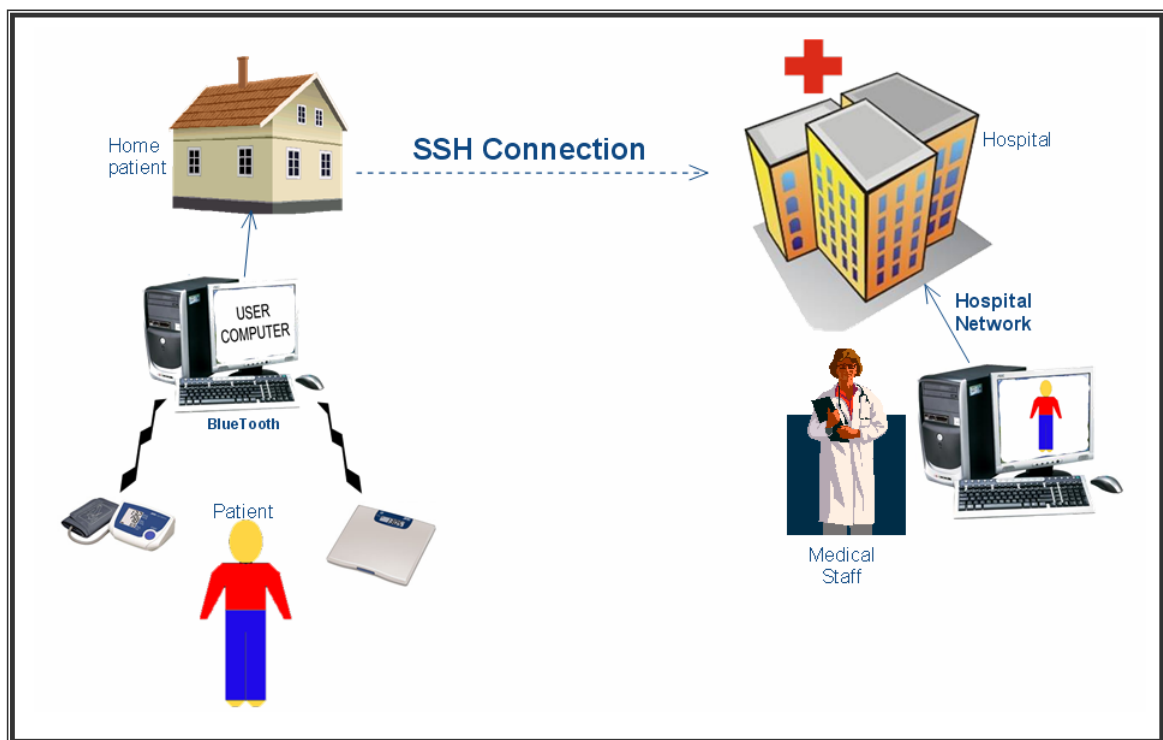


Figure 1 – Overall system

The implemented application provides with the following features:

- 1 - The possibility to measure weight and blood pressure through a really simple and easy interface for the patient

- 2 - A local database implemented in the user's machine. This way all the measurements are saved in the patient computer.
- 3 - Secure Communication between the local application suite and remote server. The data communication is encrypted.
- 4 - A remote database implemented in a machine at the health center's. This way the measurements from all the patients under medical supervision are stored safely.
- 5 - Authenticated access for the clinical staff to the measurements on the electronic patient record.
- 6 - Data encryption: the data is stored in the local and the remote databases encrypted.
- 7 - Standarized file storing: the measurements are saved in EDF format and the EPR are saved and displayed according the HL7

1.1 Renal Failure and Home-monitoring

Renal failure is a condition manifested on millions of persons and it can be an indication of several different disorders, like for instance Nephrotic Syndrome. The most common symptom in Nephrotic Syndrome is the excess of fluid in the body, thus producing oedema formations mainly in the limbs. i.e. limb swelling. Renal failure may require medication, dietary and lifestyle modification, and in acute cases dialysis.

Dialysis is a medical procedure which is used to provide an artificial replacement for lost kidney function due to renal failure. Hemodialysis (HD) is the most common type of dialysis and patients under HD, approximated 1.5 millions worldwide, must go to a dialysis clinic on a fixed schedule multiple times per week. This kind of treatment is very disgusting and unpleasant due to the high number of visits. Moreover sometimes the systematic visits are not even necessary since the kidneys may be working relatively fine by them-selves. Therefore with a correct knowledge about liquid accumulation, patients would have control about their body state and schedule the dialysis visit accordingly to their renal function.

Peritoneal dialysis (PD) as opposed to HD can be done at home by the patient him-self, but always under medical supervision and routinely dialysis protocol. An important part of the protocol is focused to obtain an estimation of the current renal function to consequently adjust the dialysis settings appropriately to the renal function needs of the patient. [(González and Ruiz 2007)].

With the goal to increase the medical supervision during PD,, a new home monitoring system is being developed to supervise the dialysis treatment at home. And measurements of blood pressure and weight are part of such monitoring system

1.2 Blood Pressure

Blood pressure, strictly speaking: vascular pressure, refers to the force exerted by circulating blood on the walls of blood vessels, and constitutes one of the principal vital signs. The pressure of the circulating blood decreases as blood moves through arteries, arterioles, capillaries, and veins; the term blood pressure generally refers to arterial pressure. Arterial pressure is often measured via a sphygmomanometer, which uses the height of a column of mercury to reflect the circulating pressure. Although many modern vascular pressure devices no longer use mercury, vascular pressure values are still universally reported in millimetres of mercury (mmHg).

The systolic arterial pressure is defined as the peak pressure in the arteries, which occurs near the beginning of the cardiac cycle; the diastolic arterial pressure is the lowest pressure (at the resting phase of the cardiac cycle). The average pressure during the cardiac cycle is reported as mean arterial pressure; the pulse pressure reflects the difference between the maximum and minimum pressures measured.

Typical values for a resting, healthy adult human are approximately 120 mmHg (16 kPa) systolic and 80 mmHg (11 kPa) diastolic (written as 120/80 mmHg, and spoken as "one twenty over eighty") with large individual variations. These measurements of arterial pressure are not static, they experience natural variations from one heartbeat to another and throughout the day (in a circadian rhythm). They also change in response to stress, nutritional factors, drugs, or disease. Hypertension refers to arterial pressure being abnormally high, as opposed to hypotension, when it is

abnormally low. Along with body temperature, blood pressure measurements are the most commonly measured physiological parameters.

Arterial pressures can be measured invasively (by penetrating the skin and measuring inside the blood vessels) or non-invasively. The non-invasive measurements are not only simpler and quicker than invasive measurements, they require far less expertise in fitting and have virtually no complications. On top of that they are less unpleasant and painful for the patient. However, non-invasive measures may yield somewhat lower accuracy and small systematic differences in numerical results. Non-invasive measurement methods are more commonly used for routine examinations and monitoring.

Oscillometric measurements requires less skill than the auscultatory technique, and may be suitable for use by untrained staff and for automated patient home monitoring. The cuff is inflated to a pressure initially in excess of the systolic arterial pressure, and then reduces to below diastolic pressure over a period of about 30 seconds. When blood flow is nil (cuff pressure exceeding systolic pressure) or unimpeded (cuff pressure below diastolic pressure), cuff pressure will be essentially constant. It is essential that the cuff size is correct: undersized cuffs may yield too high a pressure, whereas oversized cuffs yields too low a pressure. When blood flow is present, but restricted, the cuff pressure, which is monitored by the pressure sensor, will vary periodically in synchrony with the cyclic expansion and contraction of the brachial artery, i.e., it will oscillate. Once the values of systolic and diastolic pressure are computed, not actually measured from the raw data but using an algorithm, then the results are displayed [(Ruben 2006)].

1.3 Body weight

Although many people prefer the less-ambiguous term body mass, the term body weight is overwhelmingly used in daily English speech and in biological and medical science contexts to describe the mass of an organism's body. [(Graham87 2008)] Since the total mass of the body is water, an increment or decrement in the total water contents of the body causes a change in the BW of the paint. Therefore BW in can be used as indicator of body fluid accumulation or Dehydration. [(JForget 2008)].

1.4 E-health System Overview

The main components of the system are the following:

- 1 - Bluetooth Weight & Blood Pressure devices.
- 2 - Software application in the PC of the patient's home.
- 3 - Local database and EDF format file storage, with encryption.
- 4 - Data communication with hospital server via SSH connection encapsulated in XML format.
- 5 - Remote storage in encrypted database.
- 6 - Web server application to access the electronic patient record according to HL7.

The system is divided in 4 main functional blocks:

1. Databases and information access.
2. System security
3. Electronic clinical reports
4. Management software.

Each of these functional blocks are described in the following sections of this thesis report, explaining in detail the implementation of each block and the criteria of choice for the database, communication protocol, file formats, and other important functional parts of the E-Health application..

2. DATABASE AND INFORMATION ACCESS

Our system consists of two different databases. One local located in the user's machine which just store information about one patient and another one located in the server machine where all patient's data is stored. Both are implemented with a Relational DataBase Management System (RDBMS) format which data is stored in the form of tables and the relationship among the data is also stored in the form of tables.

2.1 System Database

A SQL engine has been selected to manage and store information, more specifically MySQL and using Open DataBase Connectivity (ODBC) we can communicate our application with our MySQL database.

The MySQL engine have been chosen according to the following criteria and reasoning [(MySQL_Staff 2008)]:

1. **Scalability and Flexibility**: The MySQL database server supports the capacity to handle deeply embedded applications with a footprint of only 1MB for running massive data warehouses holding terabytes of information. Platform flexibility is almost the main feature of MySQL supporting all versions of Linux, UNIX, and Windows being supported. And, of course, the open source nature of MySQL, that allows complete customization for our database server.
2. **High Performance**: Whether our application, MySQL is a high-speed transactional processing system or a high-volume web site that services a billion queries a day. With high-speed load utilities, distinctive memory caches, full

text indexes, and other performance-enhancing mechanisms, MySQL offers us the right services for any system.

3. **Availability and Reliability:** Rock-solid reliability and constant availability are hallmarks of MySQL.
4. **Robust Transactional Support:** MySQL offers a powerful transactional database engine. Features include complete ACID (atomic, consistent, isolated, durable) transaction support, unlimited row-level locking, distributed transaction capability, and multi-version transaction support where readers never block writers and vice-versa. Full data integrity is also assured through server-enforced referential integrity, specialized transaction isolation levels, and instant deadlock detection.
5. **Strong Data Protection:** MySQL offers exceptional security features that ensure absolute data protection. In terms of database authentication, MySQL provides powerful mechanisms for ensuring only authorized users have entered to the database server, with the ability to block users down to the client machine level being possible. SSH and SSL support are also provided to ensure safe and secure connections. This paragraph is very important for this application because the transactions are done with patient confidential information and security mechanisms must be implemented.
6. **Easy Management:** MySQL offers exceptional quick-start capability with the short time from software download to installation. MySQL also provides a complete suite of graphical user interface for management, troubleshooting, and controlling the operation of the MySQL servers from a single workstation.
7. **Open Source Freedom and 24 x 7 Support:** MySQL is not a typical open source project. All the software is owned and supported by MySQL AB, and because of this, a unique cost and support model are available that provides a unique combination of open source freedom and trusted software with support.

2.1.1 Patient Local Database

The patient database structure is as follows:

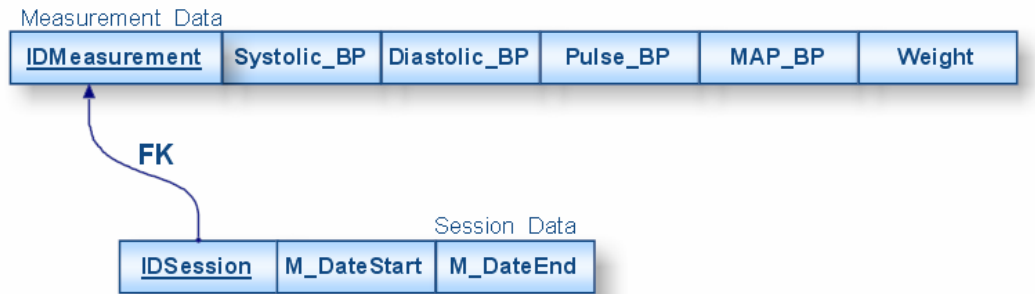


Figure 2 – Patient database structure

Measurement Data table

- IDMeasurement – INT UNSIGNED AUTO_INCREMENT PRIMARY KEY.
- Systolic_BP – INTEGER field with systolic pressure.
- Diastolic_BP – INTEGER field with diastolic pressure.
- Pulse_BP – INTEGER field with pulse rate.
- MAP_BP – INTEGER field with mean arterial pressure.
- Weight – INTEGER field with patient weight.

Session Data table

- IDSession – INT UNSIGNED AUTO_INCREMENT PRIMARY KEY.
- M_StartDate – DATETIME field with start date measurement.
- M_EndDate – DATETIME field with end date measurement.

2.1.2 Server Database

The server database structure is as follows:

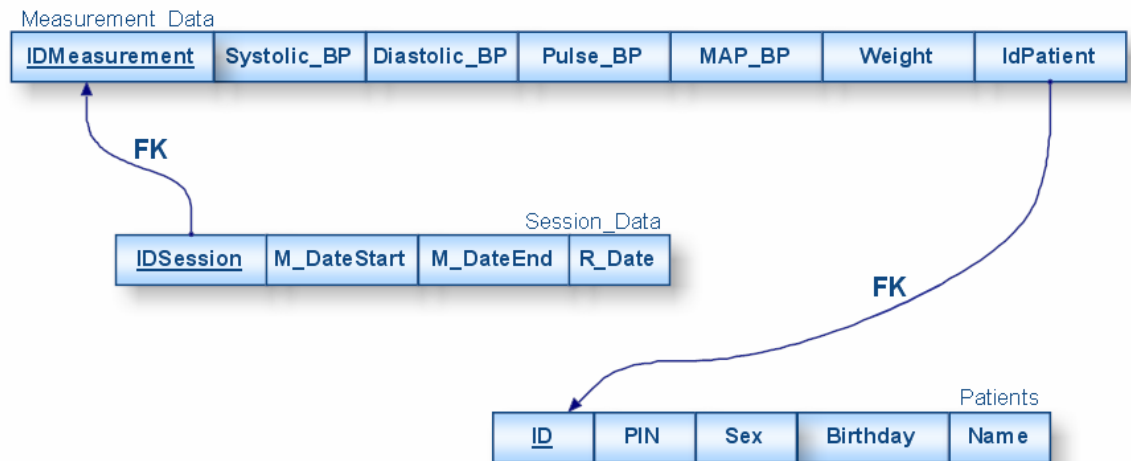


Figure 3 – Server database structure

Measurement_Data table

- IDMeasurement – INT UNSIGNED AUTO_INCREMENT PRIMARY KEY.
- Systolic_BP – INTEGER field with systolic pressure.
- Diastolic_BP – INTEGER field with diastolic pressure.
- Pulse_BP – INTEGER field with pulse rate.
- MAP_BP – INTEGER field with mean arterial pressure.
- Weight – INTEGER field with patient weight.

Session_Data table

- IDSession – INT UNSIGNED AUTO_INCREMENT PRIMARY KEY.
- M_StartDate – DATETIME field with start date measurement.
- M_EndDate – DATETIME field with end date measurement.
- R_Date – DATETIME field with reception time on server.

Patients table

- ID – INT UNSIGNED AUTO_INCREMENT PRIMARY KEY.
- PIN – VARCHAR(50) field with patient's personal identification number.
- Sex – CHAR field with patient's sex.
- Birthday – DATETIME field with patient's birth date.
- Name – VARCHAR(50) field with patient's name.

2.2 System Overview and Information Access

Next figure shows a brief overview of the system and information access.

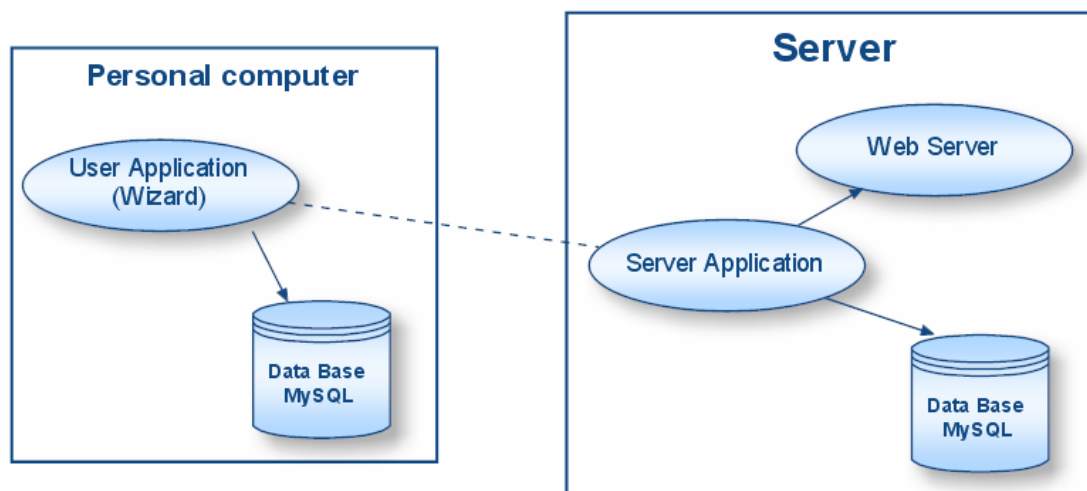
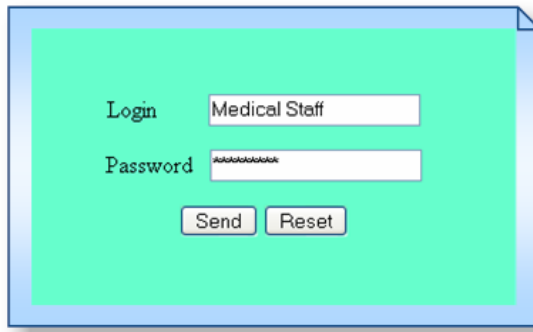


Figure 4 – Overall vision system

The web server has been design for medical staff to access the Electronic Patient Clinical Reports (EPR) via a web interface. The web interface is built by JavaServer Pages. JSP is a Java technology that allows software developers to dynamically generate HTML, XML or other types of documents in response to a Web client request. The technology allows Java code and certain pre-defined actions to be embedded into static content.

Web pages format have been defined as follows:

- **index.jsp** – This page is used to log medical staff. It shows login field and password field and send the information written to redirect.jsp

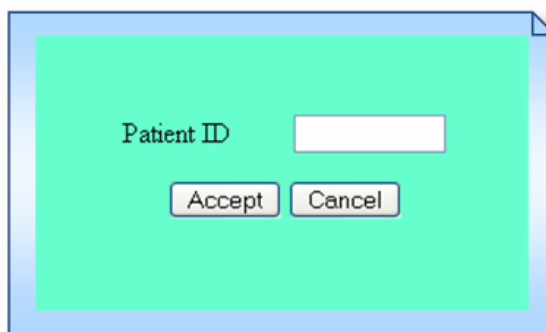


The screenshot shows a web browser window with a light blue border. The background is light green. There are two text input fields. The first is labeled 'Login' and contains the text 'Medical Staff'. The second is labeled 'Password' and contains a series of asterisks. Below the password field are two buttons: 'Send' and 'Reset'.

index.jsp

Figure 5 – Web page “index.jsp”

- **redirect.jsp** – This site doesn’t show anything to the user. It is just to check if medical staff login and password are correct. In case to be correct, the page redirect to IDRequest.jsp, otherwise it redirect to index.jsp
- **IDRequest.jsp** – This site is only accessible after authentication. It shows a ID Patient field where medical staff has to write the personal identification number of the patient they want to see the EPR.



The screenshot shows a web browser window with a light blue border. The background is light green. There is a text input field labeled 'Patient ID'. Below the field are two buttons: 'Accept' and 'Cancel'.

IDRequest.jsp

Figure 6 – Web page “IDRequest.jsp”

- **showReport.jsp** – This site is only accessible after authentication. In case of correct identification number the page show the CDA, otherwise it shows an error message “Non patient found with this ID”. This page takes the electronic clinical report in XML format from reports and executes a Extensible Stylesheet Language Transformations (XSLT) with cda.xsl to obtain a HTML with the EPR of the patient. Then the EPR is shown to the medical staff. The complete EPR, including graphics are shown in Software Management chapter.

Good Health Clinic Consultation note		
Consultant:	Doctor No	
Date:	April 15, 2008	
Patient:	Agustin Garcia	MRN: 123456
Birthdate:	September 25, 1975	
History of Present Illness		
Agustin Garcia, is a 32 year old male referred for further asthma management. Onset of asthma in his teens. He was hospitalized twice last year, and already twice this year.		
Blood pressure and weight measurements		
• Systolic: 100 Diastolic: 50 Pulse: 42 Weight: 70.0 Date of measurement: 2008-04-12 14:33:12 • Systolic: 105 Diastolic: 58 Pulse: 51 Weight: 69.9 Date of measurement: 2008-04-13 14:25:49 • Systolic: 127 Diastolic: 69 Pulse: 72 Weight: 70.3 Date of measurement: 2008-04-14 14:30:01		
Past Medical History		
• Asthma • Hypertension • Osteoarthritis, right knee		
Medications		
• Theodor 200mg BID • Proventil inhaler 2puffs QID PRN • Prednisone 20mg qd • HCTZ 25mg qd		
Allergies		
• Penicillin - Hives • Aspirin - Wheezing		
Social History		
• Smoking :: 1 PPD between the ages of 20 and 55, and then he quit. • Alcohol :: rare		
Physical Examination		
• Vital Signs :: BP 118/78; Resp 16 and unlabored; T 98.6F; HR 86 and regular • Skin :: Erythematous rash, palmar surface, left index finger. • Lungs :: Clear with no wheeze. Good air flow. • Cardiac :: RRR with no murmur, no S3, no S4.		
Labs		
• CXR 02/03/1999: Hyperinflated. Normal cardiac silhouette, clear lungs. • Peak Flow today: 260 l/m		
Assessment		
• Asthma, with prior smoking history. Difficulty weaning off steroids. Will try gradual taper. • Hypertension, well-controlled. • Contact dermatitis on finger.		
Plan		
• Complete PFTs with lung volumes. • Chem-7 • Provide educational material on inhaler usage and peak flow self-monitoring. • Decrease prednisone to 20qOD alternating with 18qOD. • Hydrocortisone cream to finger BID. • RTC 1 week.		

Figure 7 – Web page “showReport.jsp”

The Web application flowchart is as follows:

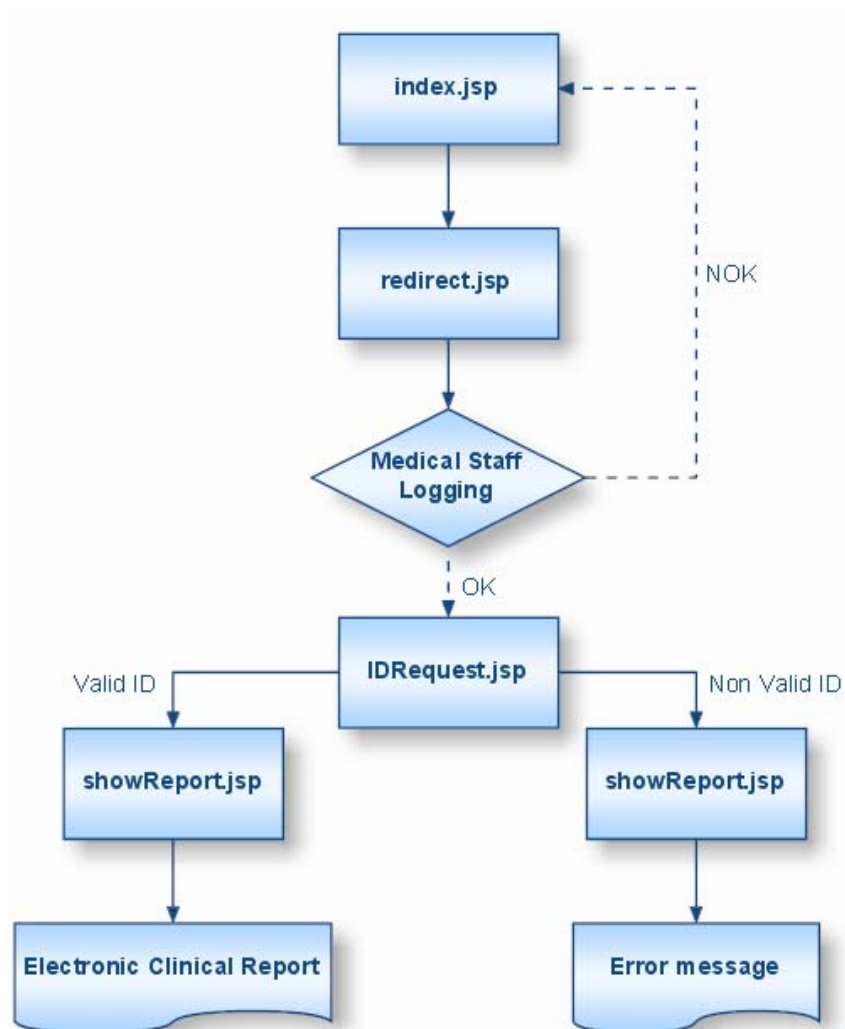


Figure 8 – Web application flowchart

3. SYSTEM SECURITY

Since storage of data contains the measurements of weight and blood pressure of patients, Confidentiality and security is one of the most important elements that we has been taken in account in the implementation of this E-health application. Therefore

3.1 EDF Files

In the user's machine, information is stored in two different places; a MySQL Database which provides its own security mechanisms and in EDF+ format files. The European Data Format (EDF) is a simple and flexible format for exchange and storage of multichannel biological and physical signals. The version used in this application is EDF+, developed in 2002 and is largely compatible to EDF, but can also contain interrupted recordings, annotations, stimuli and events. Therefore, EDF+ can store any medical recording such as EMG, Evoked potentials, ECG, as well as automatic and manual analysis results such as delta plots, QRS parameters and sleep stages. In our case the storage of our information is easier because we need to save just weight and blood pressure measurements. The EDF+ file is storaged in ASCII and it contain a file format header that contains information about the EDF format file, the patient, the recording contained in the file, the type and length of the measurements, number of channels: sampling rate, etc. After the format header all measurement data is stored in ASCII. Next figure shows in detail the EDF+ format file.

EDF HEADER RECORD	
Length	Description
8 ascii	Version of this data format
80 ascii	Local patient identification

80 ascii	Local recording identification
8 ascii	Startdate of recording (dd.mm.yy)
8 ascii	Starttime of recording (hh.mm.ss)
8 ascii	Number of bytes in header record
44 ascii	Reserved
8 ascii	Number of data records
8 ascii	Duration of a data record
4 ascii	Number of signals in data record
ns * 16 ascii	ns * label
ns * 80 ascii	ns * transducer type
ns * 8 ascii	ns * physical dimension
ns * 8 ascii	ns * physical minimum
ns * 8 ascii	ns * physical maximum
ns * 8 ascii	ns * digital minimum
ns * 8 ascii	ns * digital maximum
ns * 80 ascii	ns * prefiltering
ns * 8 ascii	ns * samples in each data record
ns * 32 ascii	ns * reserved
DATA RECORD	

Figure 9 – EDF Header

3.2 Data Encryption Standard (DES)

Since due to privacy laws about clinical data, it must be keep confidential anywhere where it is stored, EDF files are encrypted with DES because [(Kemp and Olivan 2003)]. Data Encryption Standard (DES) is a cipher, a method for encrypting information, selected as an official Federal Information Processing Standard (FIPS) for the United States in 1976

DES is the archetypal block cipher, an algorithm that takes a fixed-length string of plaintext bits and transforms it through a series of complicated operations into another cipher text bit string of the same length. In the case of DES, the block size is 64 bits. DES also uses a key to customize the transformation, so that decryption can supposedly only be performed by those who know the particular key used to encrypt. The key ostensibly consists of 64 bits; however, only 56 of these are actually used by the algorithm. Eight bits are used solely for checking parity, and are thereafter discarded. Hence the effective key length is 56 bits, and it is usually quoted as such.

There are 16 identical stages of processing, termed rounds. Before the main rounds, the block is divided into two 32-bit halves and processed alternately; this criss-crossing is known as the Feistel scheme. The Feistel structure ensures that decryption and encryption are very similar processes. The only difference is that the subkeys are applied in the reverse order when decrypting. The rest of the algorithm is identical. This greatly simplifies implementation, particularly in hardware, as there is no need for separate encryption and decryption algorithms [(Tlomak 2008)]. The Feistel structure is described in the following figure:

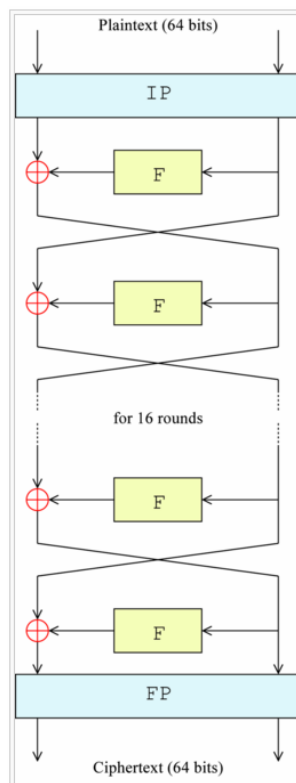


Figure 10 – Feistel structure of DES

The red + symbol denotes the exclusive-OR (XOR) operation. The F-function scrambles half a block together with some of the key. The output from the F-function is then combined with the other half of the block, and the halves are swapped before the next round. After the final round, the halves are not swapped; this is a feature of the Feistel structure which makes encryption and decryption similar processes [(Barrett, Silverman et al. 2001)].

3.3 Secure Shell (SSH)

To be consistent with the level of data confidentiality, the implemented security is not only limited to the data storing at patient's machine, the client to server communication it is also secured. To obtain a secured data communication it is needed: 1st to encrypt messages in the source prior to transmission and 2nd decrypt the received data in the destination. .

To implement this secure communication the client and server applications must encrypt and decrypt the data with a common key only known by each of them [(Barrett, Silverman et al. 2001)].

To implement secure communications there are several different protocols:

- SSH: Used instead of Telnet
- SSL: Mainly used for hypertext but also support other protocols.
- TLS: Quite similar to SSL
- HTTPS: Only used for hypertext communications.

This application uses SSH protocol. Secure Shell or SSH is a network protocol that allows data to be exchanged over a secure channel between two computers. Encryption provides confidentiality and integrity of data. SSH uses public-key cryptography to authenticate the remote computer and allow the remote computer to authenticate the user, if necessary.

SSH is typically used to log into a remote machine and execute commands, but it also supports tunnelling, forwarding arbitrary TCP ports and X11 connections; it can transfer files using the associated SFTP or SCP protocols.

An SSH server, by default, listens on the standard TCP port 22. An SSH client program is typically used for establishing connections to an SSHD daemon accepting remote connections. Both are commonly implemented on most modern operating systems, including Mac OS X, Linux, FreeBSD, Solaris and OpenVMS. Proprietary,

freeware and open source versions of various levels of complexity and completeness exist.

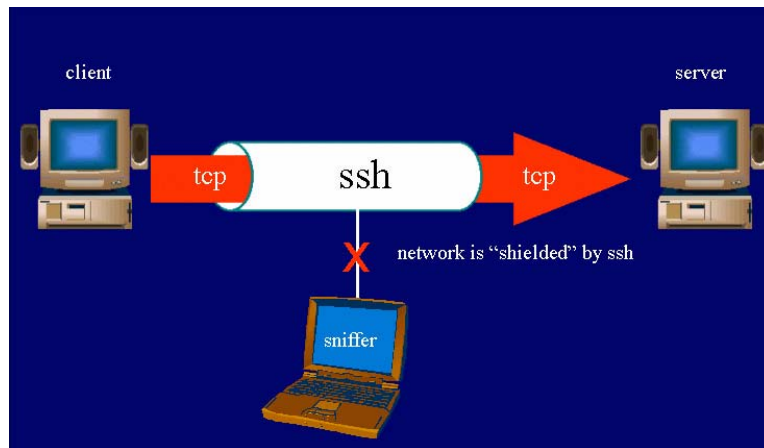


Figure 11 – SSH Security system

4. *ELECTRONIC PATIENT RECORD*

The management of health services through telematic services requires standardization of messages, formats, encoding and structure of medical records to allow interoperability between different health information systems.

For the development of a new standard to define an electronic clinical report there are several issues to pay attention to:

- Contents and structure standards (architecture).
- Clinical data representation (encoding).
- Communication standard (messages format).
- Data security, confidentiality and authentication.

Nowadays there are several projects and standards focused in ruling the approach to deal with the issue mentioned above, but the following six are the most important.

- OSI (Open Systems Interconnection)
- CORBA (Common Object Request Broker Architecture)
- GEHR (Good European Health Record)
- HL7-CDA (Clinical Document Architecture)
- openEHR
- XML/Ontology

Among all the possible standard it was decided to use in this application the HL7-CDA. HL7 was selected by its singular focus on the interface requirements of the entire health care organization, while most other standard focus only on the requirements of a particular department. Moreover, on an ongoing basis, HL7 develops a set of protocols on the fastest possible track that is both responsive and responsible to its members. The group addresses the unique requirements of the already installed hospital and departmental systems, some of which use mature technologies.

HL7 learned long ago that while data can be exchanged between systems, its usefulness is compromised unless there is a shared, well defined, and unambiguous knowledge of the meaning of the data transferred. Since much of the data being transferred is coded, either by HL7 or other organizations, HL7 began a focused effort via the formation of the Vocabulary Technical Committee to organize and maintain vocabulary terms used in its messages [(HL7_Staff)].

Moreover Health Level Seven has been actively working with XML technology since the formation of the SGML/XML Special Interest Group which has been working in recommendations on use of Extensible Markup Language (XML) standards for all of HL7's platform independent healthcare specifications and developing structured document standards for healthcare.

The expanded capability of the Reference Information Model (RIM) includes the use of the Clinical Document Architecture (CDA), a model for exchanging clinical documents. Derived from the RIM, the CDA converts documents into a format which can be read by machines as well as by humans. The CDA standards being developed by HL7 can be used to represent clinical documents such as progress notes, discharge summaries, and results of physical examinations.

The CDA organizing framework can be used to ensure clear, unambiguous representation of all patient information which is input into a computer and displayed via any software adhering to the same standard. Thus, by following the HL7 CDA, any system architect will be able to design an EPR which can be transmitted over computer networks and can be automatically integrated into any other EPR written according to the HL7 CDA standard.

5. MANAGEMENT SOFTWARE

In terms of management software the application can be split in three: the original application, the user application and the server application. The first one is the point from which we have developed our final application. The second application is a wizard suite which guides the patient through the measurement protocol of weight and blood pressure and consequently sends the information from the local machine to a server, and the third one is an application running in the server side, which receives the data from patients, and stores it in remote database, updating the patients' EPR.

5.1 Communication and Measurement Suite.

From existing source code for a primitive communication application available in Visual Basic 6.0, the first step was to develop an new application in Visual Studio .NET to show measures taken from bluetooth weight scale and bluetooth blood pressure monitor.

The main goal of this step was twofold: first to get acquainted with the measurement system and its communication protocol and second to improve interface of the primitive application implementing additional navigation functionalities. This way , the resulting application can display weight and blood pressure measurements simultaneously and it is possible to navigate along measures that had been taken previously. This navigation functionality was obtained by using an SQL Server to saved all measurement data.

To update from 6.0 to .NET, Microsoft Visual Studio .NET was used to migrate the application. The migration result from the automatic conversion tool was acceptable

but several problems appeared with the management of COM port. The way to solve these problems was to delete COM port application created by the migration tool and create a new COM port application in .NET which specifications, configurations and operation were different.

The Communication Setting panel allows the user to configure the COM port, Baud rate, parity, etc. The setting panel is as follows.

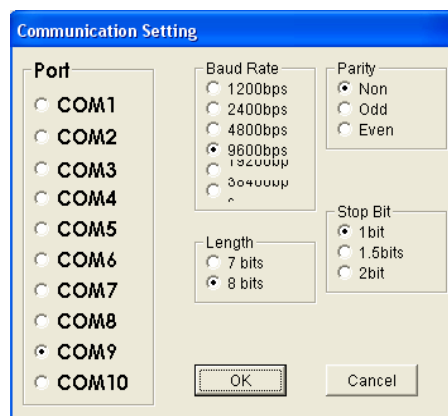


Figure 12 – Configuration COM panel

The main panel shows the weight and blood pressure measurements. Once the COM port configuration has been set, the measurements are received through the Bluetooth serial port and then they are displayed on the panel. In the next figure an example of main panel with measurements is shown.

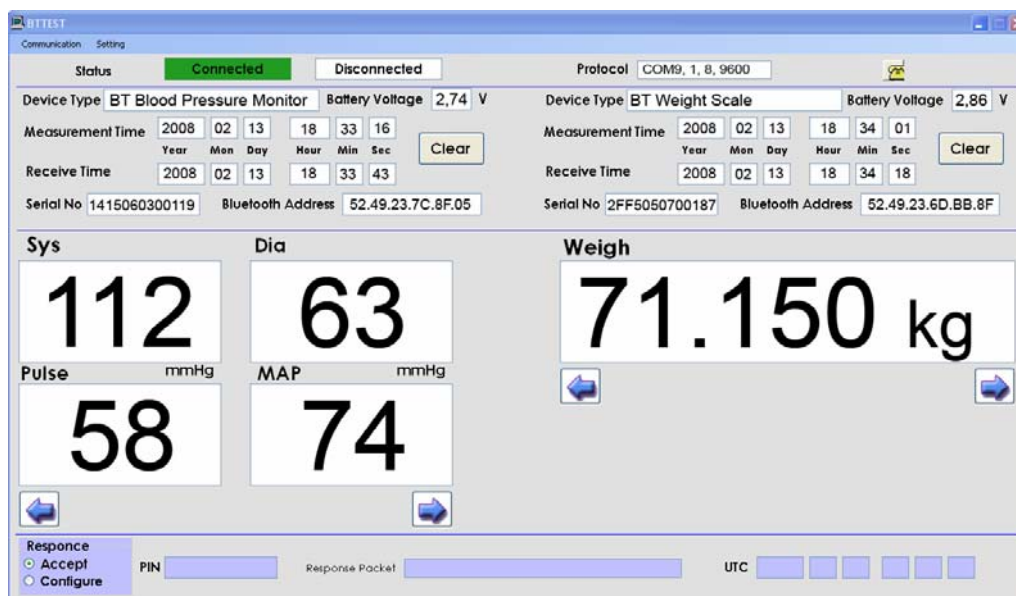


Figure 13 – A&D application modified

In the figure it is possible to see how all information from the measurements is displayed. Information like the measurement time, the measurement received time, serial number of measurement devices and even the Bluetooth Address of them.

Arrows are located under the measurement display, allowing the user to navigate through history of measurements displaying each moment, complete information about that specific measurement.

5.2 Wizard Suite

From here onwards the description focuses on the Wizard suite. This software tool is intended for patients at home. The wizard uses a user machine to measure and transmit the measurements and a server application to receive and process data from patients in server side.

The usage cases for patients, medical staff and administrator are as follows:

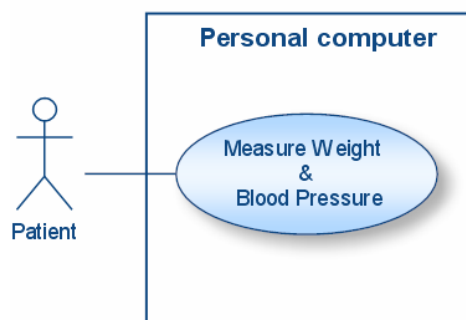


Figure 14 – Use case for patient

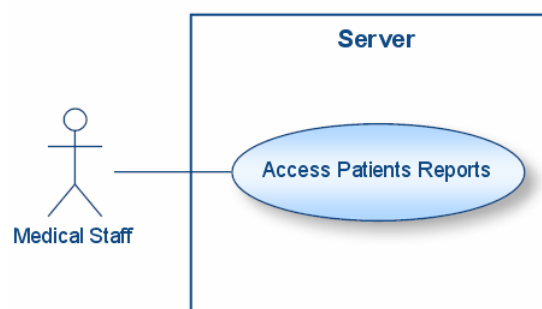


Figure 15 – Use case for medical staff

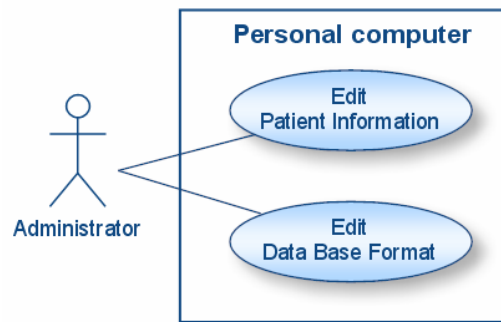


Figure 16 – Use case for administrator in user machine

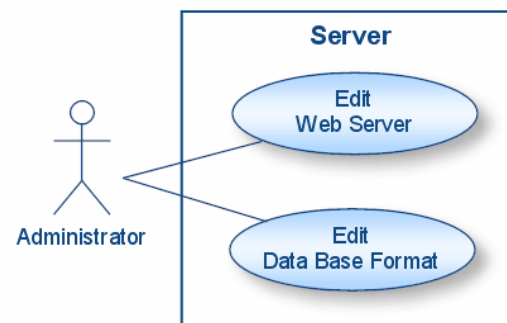


Figure 17 – Use case for administrator in server machine

5.3 System architecture

In order to have an overview of the system, the skeleton of the application is divided in different subsystems, which are responsible of a specific functional part of the system. Each subsystem interacts, itself, with others subsystems in a unidirectional or bidirectional way as necessary in each case.

Based on this architecture, the subsystems defined previously in the general architecture of the system will be explained in detail in the following sections.

In the following figure, we can see a High-level overview of the complete system.

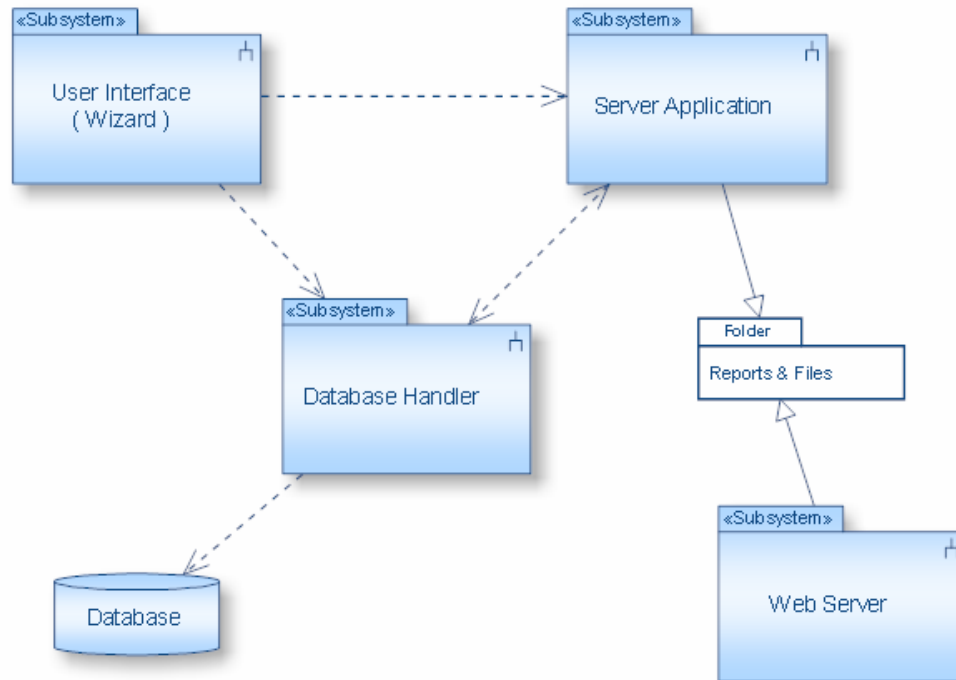


Figure 18 – High-level overview of system.

5.4 Patient's Side

The diagram in the following figure contains connections between the different components in the patient's side.

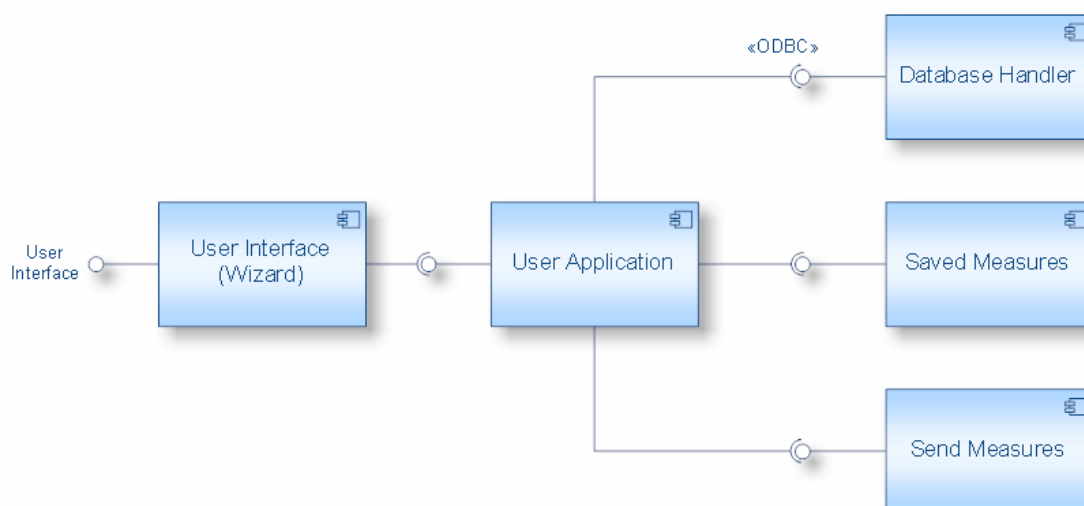


Figure 19 – Connection component diagram at the patient's side.

The User Interface subsystem is responsible of data measurements, transmission and locally stored measurements, and it is built up by the following subsystems: Data Measurement, Save Measurements and the Transmission Measurement subsystem.

The Data Measurement subsystem is a wizard which guide the user through this process, the Saved Measurements subsystem is responsible to save data locally the data and the Transmission Measurement subsystem connects the patient's application with the server through and SSH connection and sends the measurements

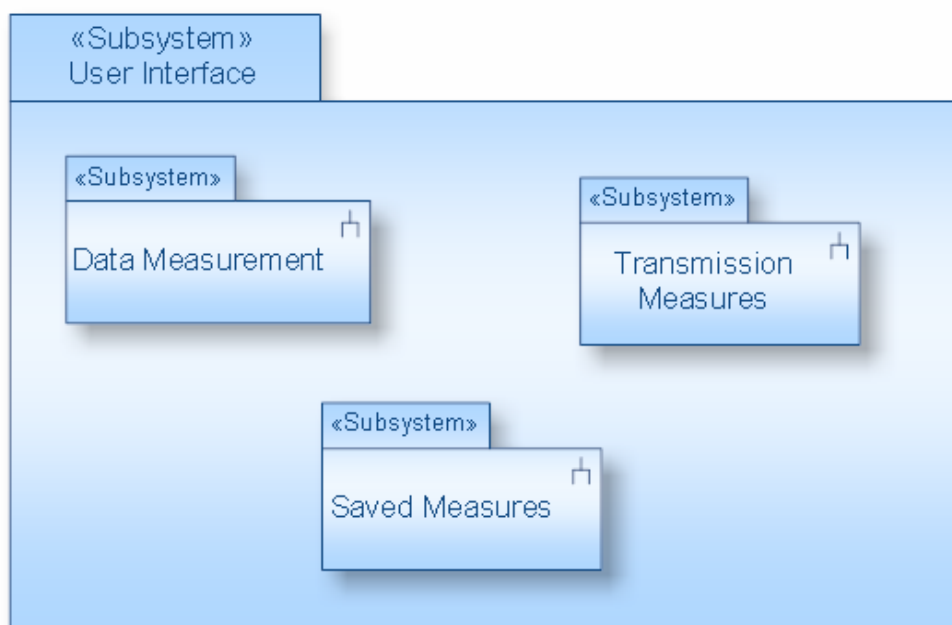


Figure 20 – User Interface subsystem

In the following sections each of the components of the subsystems and the connection diagrams are described with more detail. The application source code is also explained to facilitate understanding internal operation of the subsystems.

5.4.1 User Interface

This application has been developed to support the patient to follow the measurement protocol as indicated by the physician in charge.

In the case of Peritoneal Dialysis as part of the treatment, the patient's weight and blood pressure have to be carefully checked by appropriately qualified medical staff in order to assess on the effectiveness of the treatment.

But renal patients are not the only who need to check their weight and blood pressure regularly. People who have suffered a Heart Failure, high risk pregnancy, cerebrovascular accident, hyperlipemia, etc. also need to have a continued care which involves measurements of weight and blood pressure.

In order to facilitate as much as possible weight and blood pressure measurements and assuming that the application will be installed in user's machine at the patient's home, a very intuitive and simple graphic interface has been developed. With such simple graphical interface the risk of mistakes by the patient is reduced to a very large extent. In the following figures and sequentially the different steps of the Wizard interface are shown and explained:



Figure 21 – First patient's Wizard panel

The first panel, see figure above, appears when the patient starts the application. It just explains that the wizard will guide the user along the measurement process and that the patient should have weight scale and blood pressure monitor near by. After that the patient should click on the button 'NEXT' to proceed with the next panel.

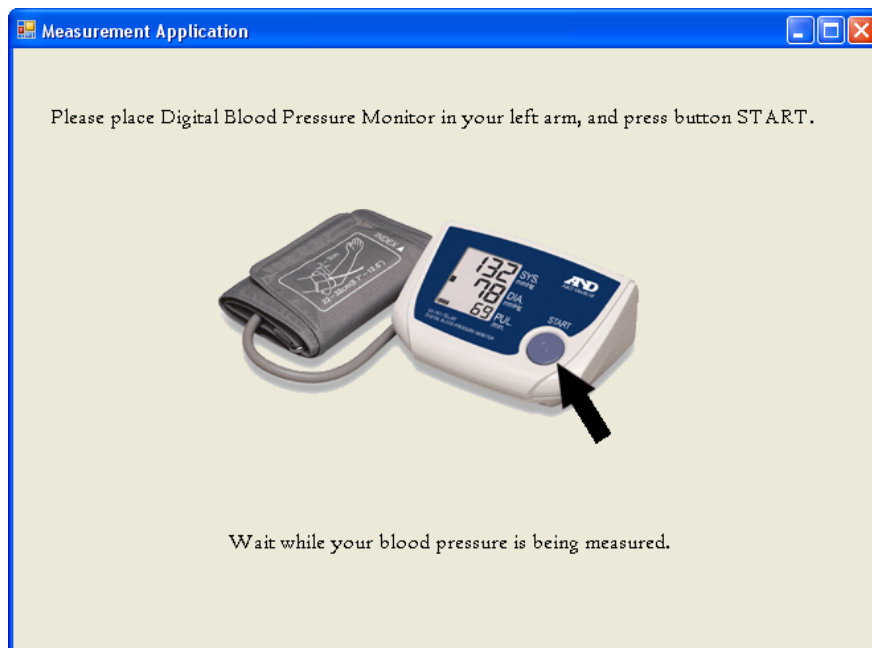


Figure 22 – Second patient’s Wizard panel

The second panel, see figure above, appears after the patient has clicked the button “NEXT” in the first panel, that means that the patient is ready to start with the measurements. The animated image shows where the patient has to press to measure his blood pressure.



Figure 23 – Third patient’s Wizard panel

After blood pressure is measured, the application continues automatically to the next wizard panel. The third panel, see figure in previous page shows to the patient where to press in the weight scale to prepare the scale for measurement and notifies him to wait until the screen shows 0,0 to measure

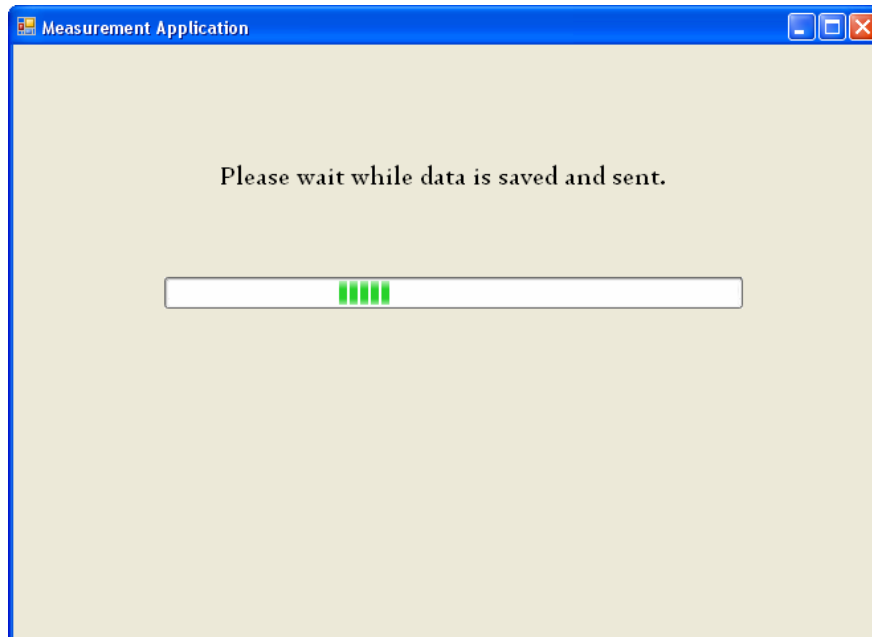


Figure 24 – Fourth patient's Wizard panel

When the Wizard application receives the weight information from the weight scale, it shows the forth panel, see figure above. This panel reports to the patient that data is being stored in the local machine's database and it is also being sent to the server for remote storing.

The last panel of the Wizard application reports to the patient that the information has been successfully stored and sent. The patient has just to click the button "FINISH" to exit the wizard.

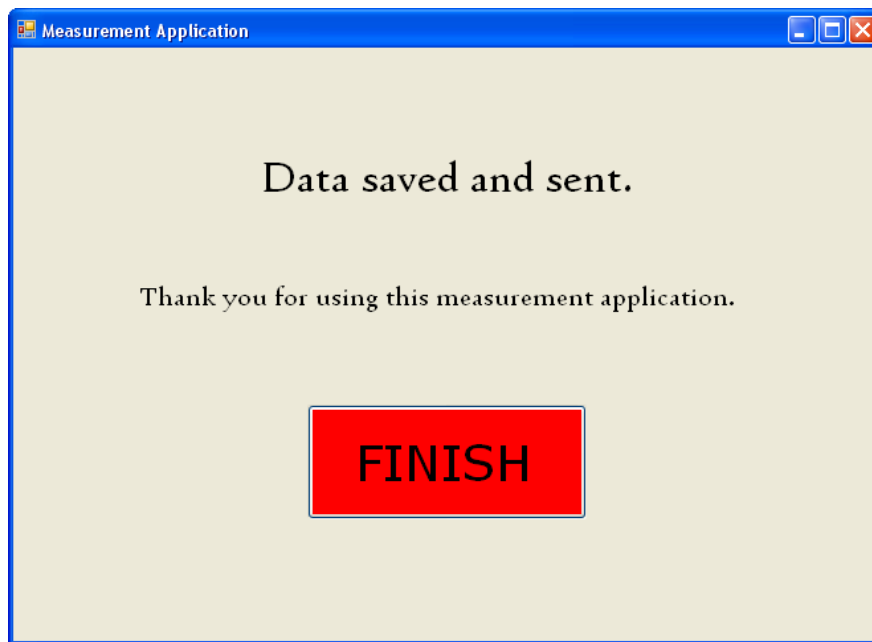


Figure 25 – Fourth patient's Wizard panel

In the case that the information from the measurements can not be sent to the server, either for an unavailable internet connection or because the application in the server is down, the program will show the following message:

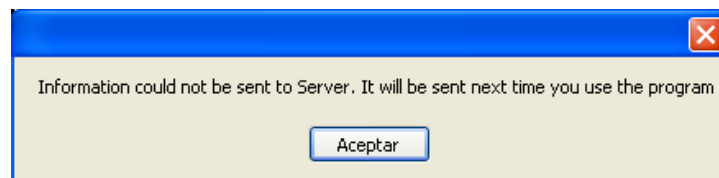


Figure 26 – Error message of patient's wizard.

IMPORTANT – Sometimes, the connection between the weight scale or blood pressure monitor with the local machine need to be updated so the PIN code is required by the local machine. This code is 39121440.

5.4.2 User Application

The patient's side application has been programmed in Microsoft Visual Studio 2005 version 8.0.50727.42. The main class, Aplicacion.vb, establishes a Bluetooth connection weight scale-local machine and blood pressure monitor-local machine. Once

the connection is established, and during the second and third Wizard's panels, the application gets all the measurement data.

Configuration parameters for patient identification are stored in a local file "dataPatient.INI". This file has to be different for each patient and the file contains the patient ID, name, sex and birth date.

Configuration parameters for the Bluetooth connection are stored in the file "BTTEST.INI". In this file the baud rate, length, parity, stop bit and port are set.

To facilitate the understanding of the different subsystems, below, the component diagrams and the code source of the main function used in the Wizard application are explained.

The Component diagrams allow the application designer to go one step further in the level of design. This diagrams are considered like an intermediate step between subsystem diagrams and class diagrams.

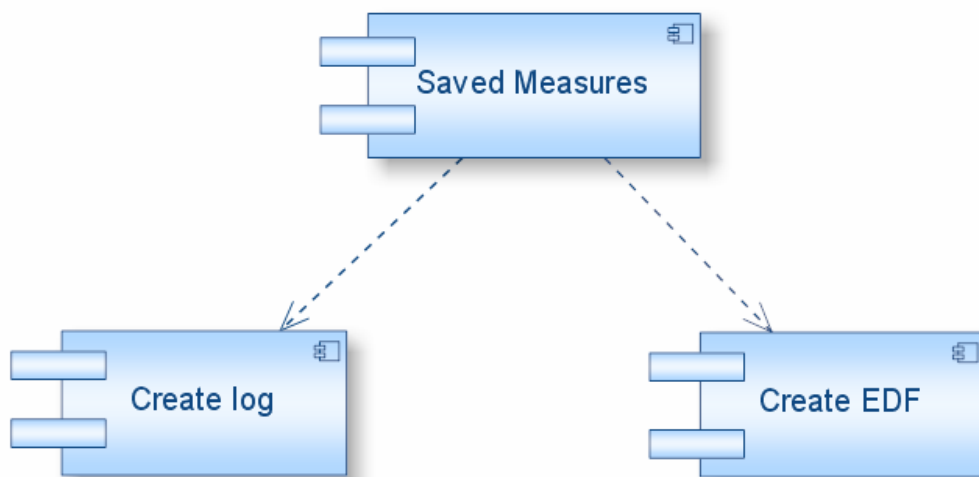


Figure 27 – Component deployment diagram for Saved Measurements subsystem.

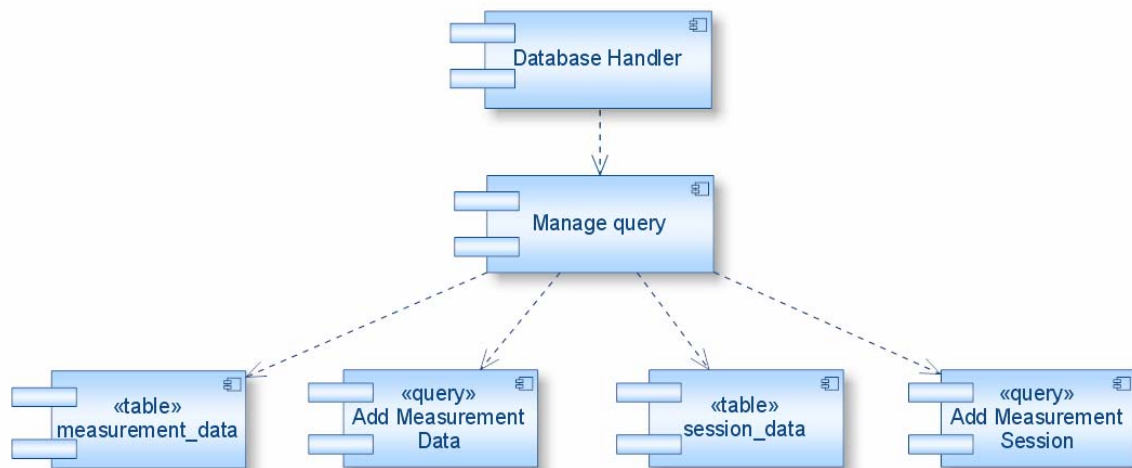


Figure 28 – Component deployment diagram for Database Handler subsystem.

After measurement time, the user application the saves data in the local database and creates an encrypted EDF file and measurement log which are also saved. To save the data, the main class calls functions located in “Save.vb”.

The main functions are called from the main application and they are as follows:

```

Sub SaveData(ByVal pesoNumerico As String, ByVal sys As Short, ByVal
    dia As Short, ByVal pulse As Short, ByVal MAP As Short,
    ByVal startTimeMeasurement As Date, ByVal
    endTimeMeasurement As Date)

    'Here me make a measurements log.
    FileOpen(1, ".\Datos\Measurements.log", OpenMode.Append)
    PrintLine(1, Str(sys) + " " + Str(dia) + " " + Str(pulse) + "
        " + Str(MAP) + " " + Str(pesoNumerico) + " " +
        startTimeMeasurement + " " + endTimeMeasurement)
    FileClose(1)

    ' Create connection
    Dim conn As MySqlConnection
    conn=New MySqlConnection(New MySqlConnectionString("localhost",
        "patient", "Login name", "password",
        3306).AsString)

    'Open connection with data base
    conn.Open()
    Dim SentenciaSql As String
    Dim instruccion As MySqlCommand
    SentenciaSql = "INSERT INTO session_data(M_DateStart,
        M_DateEnd) VALUES(' " +
        CreateDateMySQL(startTimeMeasurement)+ "',' " +
        CreateDateMySQL(endTimeMeasurement) + "',')"
    instruccion = New MySqlCommand(SentenciaSql, conn)

```

```
instruccion.ExecuteNonQuery()  
SentenciaSql = "INSERT INTO measurement_data(Systolic_BP,  
            Diastolic_BP, Pulse_BP, MAP_BP, Weight) VALUES  
            ('" + Str(sys) + "','" + Str(dia) + "','" +  
            Str(pulse) + "','" + Str(MAP) + "','" +  
            Str(pesoNumerico) + "')"   
instruccion = New MySQLCommand(SentenciaSql, conn)  
instruccion.ExecuteNonQuery()
```

End Sub

Source Code 1 – Sava_Data function.

This function creates “Measurement.log”, a log file which is stored in local machine just with the weight, systolic, diastolic, pulse and MAP information that is passed to the function through value parameters. Just below, using “MySQLDriverCS.dll” establishes a database connection, creating queries and executing them.

After creating a log and saving the data in the local database, the application creates the encrypted EDF file. At first, a plain text EDF file is created in a temporal folder and after this, the function “EncryptFile” is invoked.

```
Sub EncryptFile(ByVal sInputFilename As String, ByVal sOutputFilename  
            As String)  
  
    Const sSecretKey As String = "Password"  
    Dim fsInput As New FileStream(sInputFilename, FileMode.Open,  
            FileAccess.Read)  
    Dim fsEncrypted As New FileStream(sOutputFilename,  
            FileMode.Create, FileAccess.Write)  
    Dim DES As New DESCryptoServiceProvider()  
  
    'Establish secret key for DES algorithm.  
    'It is necessary a 64 bit key  
    DES.Key = ASCIIEncoding.ASCII.GetBytes(sSecretKey)  
    'Establish the initialization vector  
    DES.IV = ASCIIEncoding.ASCII.GetBytes(sSecretKey)  
    'Create DES encryption  
    Dim desencrypt As ICryptoTransform = DES.CreateEncryptor()  
    Dim cryptostream As New CryptoStream(fsEncrypted, desencrypt,  
            CryptoStreamMode.Write)  
  
    Dim bytearrayinput(fsInput.Length - 1) As Byte  
    fsInput.Read(bytearrayinput, 0, bytearrayinput.Length)  
    'Write DES file encrypted  
    cryptostream.Write(bytearrayinput, 0, bytearrayinput.Length)  
    cryptostream.Close()
```

End Sub

Source Code 2 – EncryptFile function.

Parameter sInputFilename is the location and source file's name above, non encrypted. sOutputFilename is the location and destination file's name already encrypted with DES.

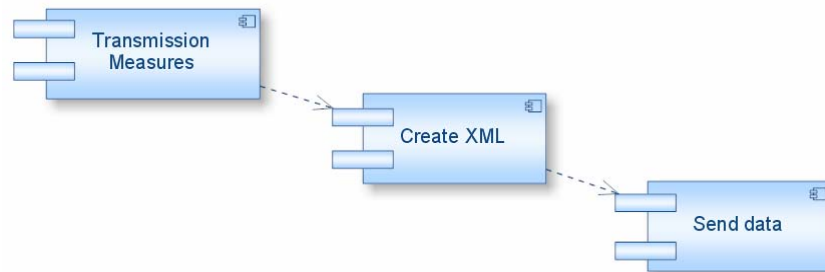


Figure 29 – Component deployment diagram for Transmission Measurements subsystem.

When all the information has been stored on the local machine, the application starts the process that creates the XML file and sends the file to the server via SSH (using Jscap library). For safety reasons, XML files contents just the ID patient number and never other fields like name, birth date, etc. An example of the produce XML files is shown below:

```

<?xml version="1.0" encoding="UTF-8" ?>
<Data>
  <MeasurementType>001</MeasurementType>

  <PersonalData>
    <PIN>51092458M</PIN>
  </PersonalData>

  <MeasurementData>
    <StartDataMeasurement>2008-02-16 14:55:8</StartDataMeasurement>
    <EndDataMeasurement>2008-02-16 14:56:26</EndDataMeasurement>
    <Systolic>108</Systolic>
    <Diastolic>62</Diastolic>
    <Pulse>49</Pulse>
    <MAP>80</MAP>
    <Weight>7130</Weight>
  </MeasurementData>
</Data>

```

Source Code 3 – XML example.

The functions that implement the creation of the XML file and its subsequent transmission to the server are CreateXML and SendData. The source code is as follows:

```

Sub CreateXML(ByVal pesoNumerico As String, ByVal sys As Short, ByVal
    dia As Short, ByVal pulse As Short, ByVal MAP As Short,
    ByVal startTimeMeasurement As Date, ByVal
    endTimeMeasurement As Date)

    Dim srLector As StreamReader = New
    StreamReader("dataPatient01.INI")
    Dim PIN, name, fileName As String
    Dim sex As Char
    Dim birthday As Date
    save = New Save
    PIN = srLector.ReadLine()
    sex = srLector.ReadLine()
    birthday = srLector.ReadLine()
    name = srLector.ReadLine()
    Dim swEscriitor As StreamWriter
    fileName = ".\tmp\" + PIN.ToString + startTimeMeasurement.
        ToShortDateString.Replace("/", "-").ToString +
        startTimeMeasurement.ToLongTimeString.Replace(":",
        "-").ToString + ".xml"
    swEscriitor = New StreamWriter(fileName)
    swEscriitor.WriteLine("<?xml version=" + Chr(34) + "1.0" +
    Chr(34) + " encoding=" + Chr(34) + "UTF-8" + Chr(34) + ">")
    swEscriitor.WriteLine("<Data>")
    swEscriitor.WriteLine("<MeasurementType>001</MeasurementType>")
    swEscriitor.WriteLine("<PersonalData>")
    swEscriitor.WriteLine("<PIN>" + PIN + "</PIN>")
    swEscriitor.WriteLine("</PersonalData>")
    swEscriitor.WriteLine("<MeasurementData>")
    swEscriitor.WriteLine("<StartDataMeasurement>" + save.CreateDate
        MySQL(startTimeMeasurement) +
        "</StartDataMeasurement>")
    swEscriitor.WriteLine(" <EndDataMeasurement>" + save.CreateDate
        MySQL(endTimeMeasurement) +
        "</EndDataMeasurement>")
    swEscriitor.WriteLine("<Systolic>" + Str(sys) + "</Systolic>")
    swEscriitor.WriteLine("<Diastolic>" + Str(dia) + "</Diastolic>")
    swEscriitor.WriteLine("<Pulse>" + Str(pulse) + "</Pulse>")
    swEscriitor.WriteLine("<MAP>" + Str(MAP) + "</MAP>")
    swEscriitor.WriteLine("<Weight>" + Str(pesoNumerico) + "</Weight>")
    swEscriitor.WriteLine(" </MeasurementData>")
    swEscriitor.WriteLine("</Data>")
    swEscriitor.Close()
    Dim ContadorDeArchivos As System.Collections.ObjectModel.
        ReadOnlyCollection(Of String)

    Dim sendado As Boolean
    Dim files As String()
    files = Directory.GetFiles(".\tmp\")
    ContadorDeArchivos = My.Computer.FileSystem.GetFiles(".\tmp\")
    Do
        sendado = SendData(files(ContadorDeArchivos.Count - 1))
        If (sendado = True) Then
            File.Delete(files(ContadorDeArchivos.Count - 1))
        End If
        ContadorDeArchivos=My.Computer.FileSystem.GetFiles(".\tmp\")
    Loop Until ContadorDeArchivos.Count = 0 Or sendado = False

End Sub

```

Source Code 4 – CreateXML function.

In the source code above, is really important to explain field: “MeasurementType”.

Currently the server application can receive any file regardless of the value of the field “Measurement Type” but it is only programmed to process files with weight and blood pressure measurements. It is intended that in the near future it will receive files with other type of measurement e.g. Electrical Bioimpedance, ECGs.. For Blood Pressure and Weigth the “MeasurementType” field is 001.

```
Private Function SendData(ByVal fileName As String) As Boolean

    Try
        Dim hostname As String
        hostname = "quartz.ih.utb.hb.se"

        Dim username As String
        username = "login name"

        Dim password As String
        password = "password"

        Dim ssh As SshParameters
        ssh = New SshParameters(hostname, username, password)
        ssh.Port = 25
        Dim sftp As Sftp
        sftp = New Sftp(ssh)
        sftp.LicenseKey = "SSH Factory for .NET License Key"

        sftp.Connect()
        Dim archivo As FileInfo
        archivo = New FileInfo(fileName)
        Dim pathComplete As String
        pathComplete = "Program Files (x86)/Apache Software
                        Foundation/Tomcat 5.5/webapps/Medical/
                        incomming/" + fileName.Substring(6).ToString

        sftp.Upload(archivo, pathComplete)

        sftp.Disconnect()
        Return True
    Catch ex As Exception
        MessageBox.Show("Information could not be sent to Server. It
                        will be sent next time you use the program")
        Return False
    End Try

End Function
```

Source Code 5 – SendData function.

The following flow diagram show how the Wizard application works from the beginning when patient starts the application until all the measurements are taken.

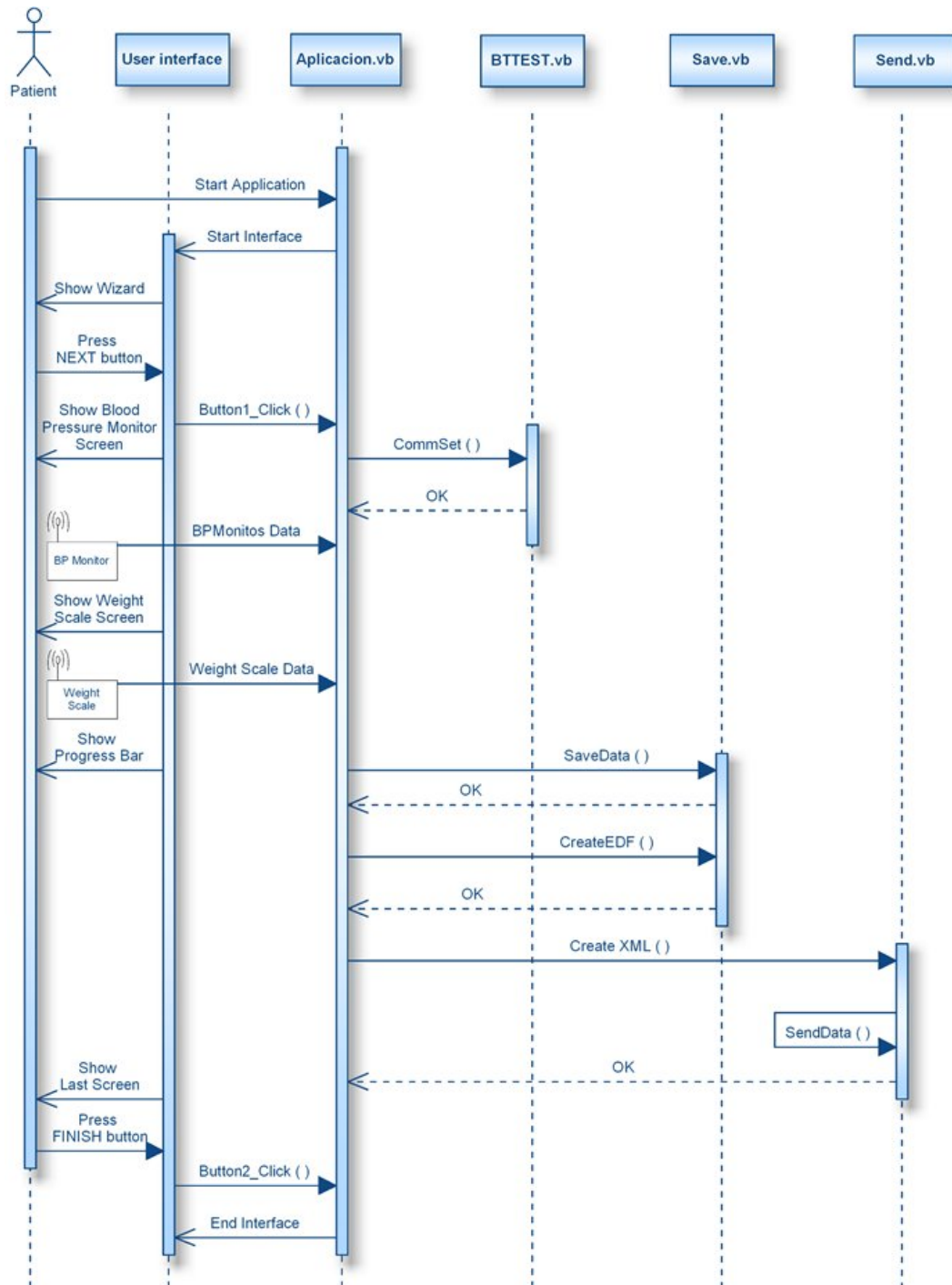


Figure 30 – Flow diagram of the Wizard application.

SendData function in Send class is the function establishes SSH connection with the server through Jscope.ssh.dll.

5.5 Server Application

On the remote server, there is a java application running. It is located in same directory that the web server and it is continually checking the incoming folder. If some file is detected in this folder, the application processes it.

Incoming files are XML extension, so the best way to process them is through an xml parser. Below, The file system server and function, contents, specification, etc. of each folder are explained. Root directory structure is as follows.

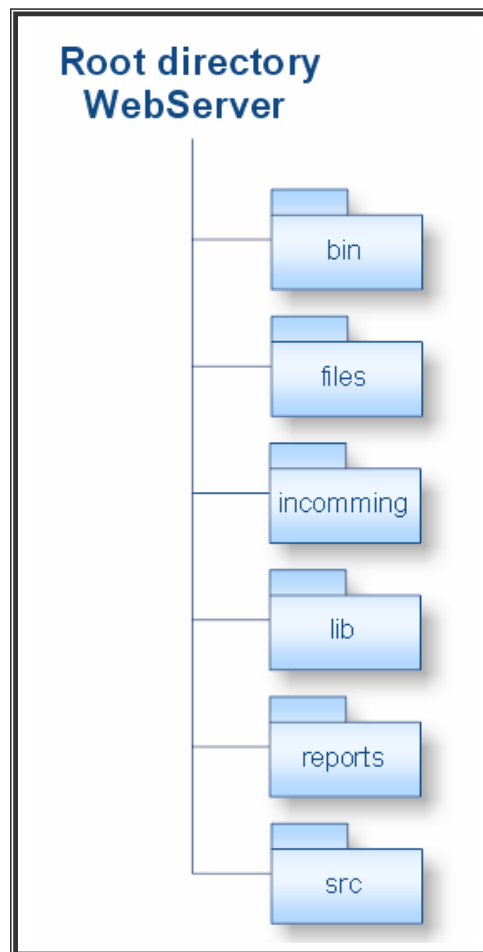


Figure 31 – Root directory organization.

The “src” folder contains the *.java files with the source code of our application. From files located in this folder, all the other paths are referenced. For an easier understanding of the application, source code and different classes will be explained below.

This class diagram shows the relationship between classes.

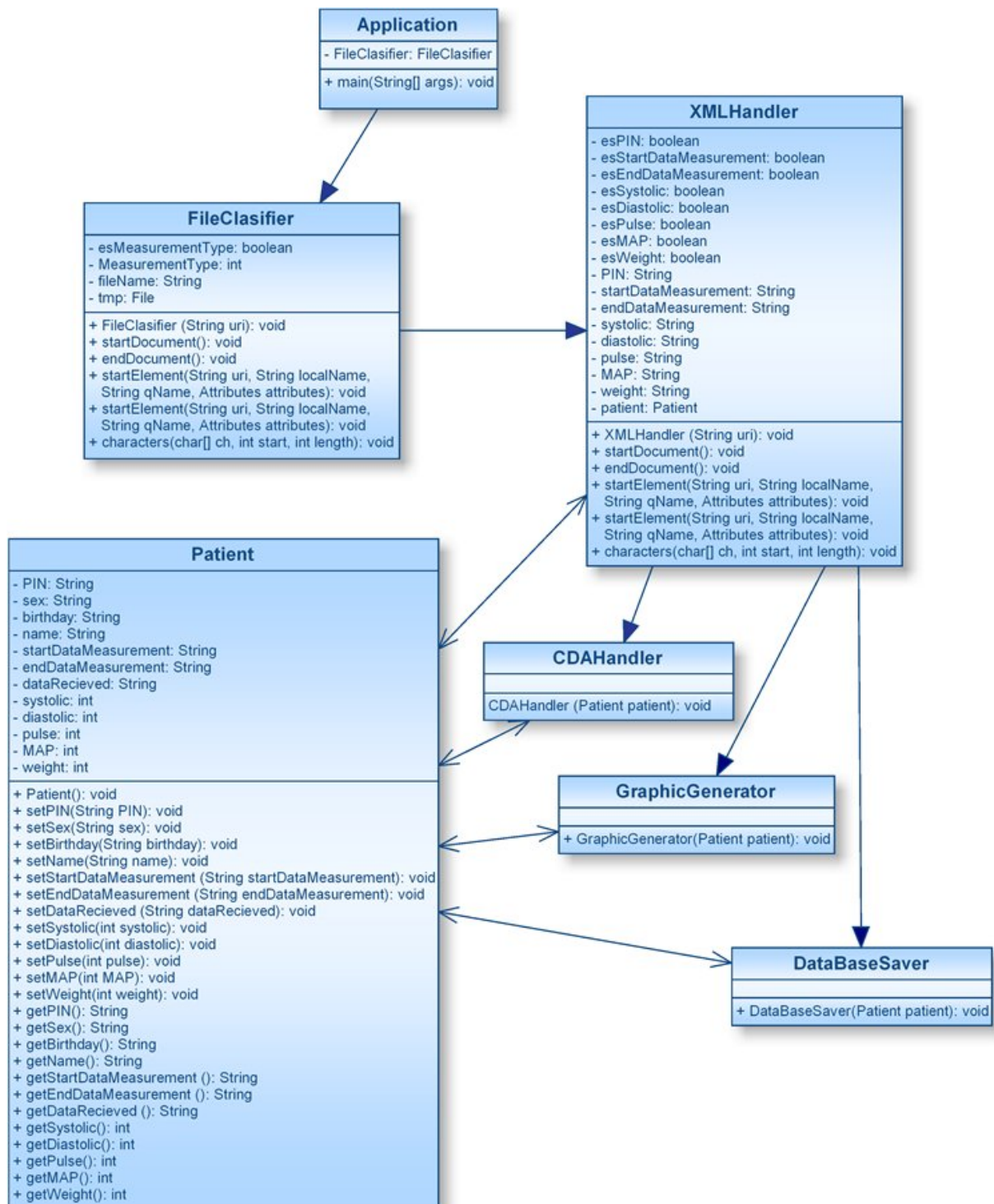


Figure 32 – Class diagram of the server application.

```
import java.io.*;

public class Application {

    /**
     * @author Agustin Garcia Perez
     * @version 1.0 beta
     */

    public static void main(String[] args){

        System.out.println("Program running");
        while(true){
            File folder = new File("./incomming");
            String[] files = folder.list();
            if(files.length != 0)
            {
                FileClassifier clasifier = new FileClassifier
                ("./incomming/" + files[0]);
                File tmp = new File("./incomming/" + files[0]);
                tmp.delete();
            }
        }
    }
}
```

Source Code 6 – Application Class.

This is the main class of our application. First it shows a message reporting that the application is running. After that it executes an infinite loop, testing “incomming” folder, where new files are uploaded from user machines. If some file is detected in this folder, a new FileClassifier object is created. Once the file is processed, it is removed from this folder.

```
import java.io.*;
import java.nio.channels.*;
import javax.xml.parsers.*;
import org.xml.sax.Attributes;
import org.xml.sax.SAXException;
import org.xml.sax.helpers.DefaultHandler;

public class FileClassifier extends DefaultHandler implements
Serializable{

    /**
     * @author Agustin Garcia Perez
     * @version 1.0 beta
     */

    private boolean esMeasurementType = false;
    private int MeasurementType = 0;
    private String fileName;
    File tmp;
```

```

public FileClassifier (String uri){
    SAXParserFactory factory = SAXParserFactory.newInstance();
    try {
        fileName=uri;
        factory.setValidating(true);
        SAXParser saxParser = factory.newSAXParser();
        saxParser.parse(uri, this);
    }catch (Throwable t) {
        t.printStackTrace();
    }
}

public void startDocument() throws SAXException{}

public void endDocument() throws SAXException{
    tmp = new File(fileName);
    if(MeasurementType==1){
        XMLHandler manejador = new XMLHandler(fileName);
    }else{
        try{
            FileInputStream fis = new FileInputStream(fileName);
            FileOutputStream fos = new FileOutputStream("./files/"
                + fileName.substring(12));
            FileChannel cFuente = fis.getChannel();
            FileChannel cDestino = fos.getChannel();
            cFuente.transferTo(0,cFuente.size(),cDestino);
            fis.close();
            fos.close();
        }catch (IOException ex) {}
    }
}

public void startElement(String uri, String localName, String
    qName, Attributes attributes) throws SAXException{
    if (qName.equals("MeasurementType")) {
        esMeasurementType = true;
    }
}

public void endElement(String uri, String localName, String
    qName) throws SAXException{}

public void characters(char[] ch, int start, int length) throws
    SAXException{
    String contenido="";
    for(int i=0;i<length;i++){
        contenido = contenido + ch[i+start];
    }
    if(esMeasurementType){
        MeasurementType = Integer.parseInt(contenido);
        esMeasurementType=false;
    }
}
}

```

Source Code 7 – FileClassifier Class.

The Builder of this class receives as parameter the file name that has to be parsed. As explained in previous pages, there is a field to identify the measurement types contained in each file: “MeasurementType”.

Since the server processes XML files to use a SAX parser is easy, powerful and intuitive. When a XML file is being parsed for the first time, the label <MeasurementType> is the first one to be checked. To differentiate between different type of measurements different codes were assigned to different files according to the containing information.

Measurement Type	Code
Weight and Blood Pressure	001
Bioimpedance	002
ECG's	003
EKG's	004
.....	

Figure 33 – Codes for measure types.

The behind choice of these codes for this different measurement types is because other projects are being developing in parallel using bioimpedance measurements. Indeed, new codes can be added in the future for another kind of measures.

The operation of this class is very simple. If MeasurementType is equal to 001, an XMLHandler object is created, otherwise the file is moved to “files” folder. XMLHandler is another SAX parser class, which in this case, extract information from the XML file and stores it in a “Patient” object. Below, an extract of Patient class is shown where we can see how the data is managed in it. Functions are omitted because of its simplicity (get and set).

```
public class Patient {  
  
    /**  
    * @author Agustin Garcia Perez  
    * @version 1.0 beta  
    */  
  
    String PIN;  
    String sex;
```

```
String birthday;
String name;
String startDataMeasurement;
String endDataMeasurement;
String dataRecieved;
int systolic;
int diastolic;
int pulse;
int MAP;
int weight;
```

Source Code 8 – Patient Class.

In the XMLHandler class, it is important to note that all fields to fill Patient object are treated with the exception of name, birth date and sex. The reason behind these fields not being transmitted is just a security reason. Otherwise, someone who gets the XML file (sniffers), would be able to link data and patient. The method we have used to solve this data omission is executing a query to database. Below we can see the source code included in the XMLHandler to ask the database for the patient's data and after the code to save the new measurement.

```
public class XMLHandler extends DefaultHandler implements
    Serializable{

    /**
     * @author Agustin Garcia Perez
     * @version 1.0 beta
     */

    public void endElement(String uri, String localName, String
        qName)throws SAXException{

        if (qName.equals("Data")) {
            DataBaseSaver saver = new DataBaseSaver(patient);
            CDAHandler manejador = new CDAHandler(patient);
            GraphicGenerator graphic = new GraphicGenerator(patient);
        }

        if (qName.equals("PIN")) {
            patient.setPIN(PIN);
            try{
                Connection conn = null;
                String login = "login";
                String pass = "password";
                Class.forName("com.mysql.jdbc.Driver");
                String url=("jdbc:mysql://localhost:3306/server");
                conn = DriverManager.getConnection(url, login, pass);
                Statement stmt = conn.createStatement();
                String query = "SELECT * FROM patients p WHERE p.PIN='"
                    + PIN + "'";
```

```

        ResultSet resultado = stmt.executeQuery(query);
        resultado.first();
        patient.setSex(resultado.getString("Sex"));
        patient.setBirthday(resultado.getString("Birthday"));
        patient.setName(resultado.getString("Name"));
        conn.close();
    }catch(Exception e){
        System.out.println(e);
    }
}
....
}

```

Source Code 9 – XMLHandler Class.

As we can see, when the <Data> label is detected, three new objects are created. DataBaseSaver, CDAHandler and GraphicGenerator.

DataBaseSaver is the class responsible for saving the data included in Patient object in the remote database. Patient object is provided as parameter and after connection to the database and authentication, the application ask for patientID with PIN. When patientID is recover from patients table, the queries are executed.

```

import java.sql.*;

public class DataBaseSaver {

    /**
     * @author Agustin Garcia Perez
     * @version 1.0 beta
     */

    public DataBaseSaver(Patient patient){
        try{
            Connection conn = null;
            String login = "login";
            String pass = "password";

            Class.forName("com.mysql.jdbc.Driver");
            String url=("jdbc:mysql://localhost:3306/server");
            conn = DriverManager.getConnection(url, login, pass);
            Statement stmt = conn.createStatement();
            //Query to ask for patientID
            String query = "SELECT * FROM patients p WHERE p.PIN='" +
                            patient.getPIN() + "'";
            ResultSet resultado = stmt.executeQuery(query);
            resultado.first();
            //Query to save session data
            String tmp = resultado.getString("Id");

```

```
query = "INSERT INTO session_data(M_DateStart, M_DateEnd,
    R_Date) VALUES ('" + patient.getStartDataMeasurement()
    + "','" + patient.getEndDataMeasurement() + "','" +
    patient.getDataRecieved() + "')";
stmt.executeUpdate(query);
//Query to save measurement data
query = "INSERT INTO measurement_data(Systolic_BP,
    Diastolic_BP, Pulse_BP, MAP_BP, Weight, IdPatient)
    VALUES ('" + patient.getSystolic() + "','" +
    patient.getDiastolic() + "','" + patient.getPulse() +
    "','" + patient.getMAP() + "','" +
    patient.getWeight()/100 + "','" + tmp + "')";
stmt.executeUpdate(query);
conn.close();
}catch(Exception e){
    System.out.println(e);
}
}
```

Source Code 10 – DataBaseSaver Class.

Since one of the main features of this E-health application is to allow teleconsultation in real time, it is very important that each time a patient measures his weight and blood pressure the EPR is immediately updated. A new record line is included in the patient's report and the figures are updated in the CDA as well.

The information format included is as follows and source code to include it also.

Systolic: 124 Diastolic: 62 Pulse: 57 Weight: 7260 Date of measurement: 2008-3-5 16:59:41

Figure 34 – Information added to the CDA.

```
import java.io.*;

public class CDAHandler implements Serializable {

    /**
     * @author Agustin Garcia Perez
     * @version 1.0 beta
     */

    public CDAHandler (Patient patient){
        try {
            FileReader fileR = new FileReader("./reports/" +
                patient.getPIN() + ".xml");
            FileWriter fileW = new FileWriter("./reports/" +
                patient.getPIN() + "copy.xml");
```

```

BufferedWriter writer = new BufferedWriter(fileW);
BufferedReader reader = new BufferedReader(fileR);
String cadena="";

do{
    cadena = reader.readLine();
    writer.write(cadena + "\n");
    if(cadena.equals("<caption>Blood pressure and weight
                    measurements</caption>"))
    {
        writer.write(reader.readLine() + "\n");
        writer.write("<item><content>Systolic: " +
            patient.getSystolic() + " Diastolic: " +
            patient.getDiastolic() + " Pulse: " +
            patient.getPulse() + " Weight: " + patient.getWeight()
            + " Date of measurement: " +
            patient.getStartDataMeasurement
                () + "</content></item>\n");
    }
}while(!cadena.equals("</levelone>"));
reader.close();
writer.close();
File firstFile = new File("./reports/" + patient.getPIN() +
                        "copy.xml");
File lastFile = new File("./reports/" + patient .getPIN() +
                        ".xml");

lastFile.delete();
firstFile.renameTo(lastFile);
firstFile.delete();
}catch (Exception e1) {
    e1.printStackTrace();
}
}
}

```

Source Code 11 – CDAHandler Class.

In the process to receive, analyze, process, save and update the CDAs and with the aim of facilitating to medical staff to view the results of their patients, several graphics have been added to the CDA. This graphics show weight and blood pressure evolution over time.

It has been necessary to use an external library (JFreeChart.1.0.9.jar) to generate this graphics. The Data used to generate the plots comes from the remote database. With PIN all the measurements from the patient a new graphic on is generated in *.jpeg format. Below it is shown the use of this library and with an example .The First part shows the database query and second part the graphic generation.

```

import org.jfree.chart.*;
import java.io.*;
import java.sql.*;
import org.jfree.chart.plot.*;
import org.jfree.data.xy.*;

public class GraphicGenerator {

    /**
     * @author Agustin Garcia Perez
     * @version 1.0 beta
     */

    public GraphicGenerator(Patient patient){
        try
        {
            Connection conn = null;
            String login = "root";
            String pass = "amazonas";
            Class.forName("com.mysql.jdbc.Driver");
            String url=("jdbc:mysql://localhost:3306/server");
            conn = DriverManager.getConnection(url, login, pass);
            Statement stmt = conn.createStatement();
            String query = "SELECT Id FROM patients p WHERE PIN='" +
                            patient.getPIN() + "'";
            ResultSet resultado = stmt.executeQuery(query);
            resultado.first();
            conn = DriverManager.getConnection(url, login, pass);
            stmt = conn.createStatement();
            query = "SELECT * FROM measurement_data m inner join
                    session_data s where m.IdPatient='" +
                    resultado.getInt("Id") + "' and
                    m.IdMeasurement=s.IdSession";
            resultado = stmt.executeQuery(query);
            resultado.first();

            XYSeries seriesWeight, seriesSystolic, seriesDiastolic;
            XYSeries seriesPulse, seriesMAP;
            double index = 0;
            XYDataset xyDataset;
            JFreeChart chart;
            seriesWeight = new XYSeries("Weight evolution");
            seriesSystolic = new XYSeries("Systolic evolution");
            seriesDiastolic = new XYSeries("Diastolic evolution");
            seriesPulse = new XYSeries("Pulse evolution");
            seriesMAP = new XYSeries("MAP evolution");
            do{
                seriesWeight.add(index, Double.parseDouble(resultado.
                    getString("Weight")));
                seriesSystolic.add(index, Double.parseDouble(resultado.
                    getString("Systolic_BP")));
                seriesDiastolic.add(index, Double.parseDouble(resultado.
                    getString("Diastolic_BP")));
                seriesPulse.add(index, Double.parseDouble(resultado.
                    getString("Pulse_BP")));
                seriesMAP.add(index, Double.parseDouble(resultado.
                    getString("MAP_BP")));
            }
        }
    }
}

```

```

        index++;
    }while(resultado.next());

    xyDataset = new XYSeriesCollection(seriesWeight);
    chart = ChartFactory.createXYLineChart("Medical
measurements", "Time", "Weight (Kg)", xyDataset,
PlotOrientation.VERTICAL, true, true, false);
    ChartUtilities.saveChartAsPNG(new File("reports/images/" +
patient.getPIN() + "Weight.jpeg"), chart, 500, 300);

    xyDataset = new XYSeriesCollection(seriesSystolic);
    chart = ChartFactory.createXYLineChart("Medical
measurements", "Time", "Systolic (mmHg)", xyDataset,
PlotOrientation.VERTICAL, true, true, false);
    ChartUtilities.saveChartAsPNG(new File("reports/images/" +
patient.getPIN() + "Systolic.jpeg"), chart, 500, 300);

    xyDataset = new XYSeriesCollection(seriesDiastolic);
    chart = ChartFactory.createXYLineChart("Medical
measurements", "Time", "Diastolic (mmHg)", xyDataset,
PlotOrientation.VERTICAL, true, true, false);
    ChartUtilities.saveChartAsPNG(new File("reports/images/" +
patient.getPIN() + "Diastolic.jpeg"), chart, 500, 300);

    xyDataset = new XYSeriesCollection(seriesPulse);
    chart = ChartFactory.createXYLineChart("Medical
measurements", "Time", "Pulse (/min)", xyDataset,
PlotOrientation.VERTICAL, true, true, false);
    ChartUtilities.saveChartAsPNG(new File("reports/images/" +
patient.getPIN() + "Pulse.jpeg"), chart, 500, 300);

    xyDataset = new XYSeriesCollection(seriesMAP);
    chart = ChartFactory.createXYLineChart("Medical
measurements", "Time", "MAP", xyDataset,
PlotOrientation.VERTICAL, true, true, false);
    ChartUtilities.saveChartAsPNG(new File("reports/images/" +
patient.getPIN() + "MAP.jpeg"), chart, 500, 300);
}catch(Exception e){}
    }
}

```

Source Code 12 – GraphicGenerator Class.

Here we can see three different graphic examples for weight, systolic and pulse measurements over time. Time line represents the measurement number since the patients started the home monitoring.

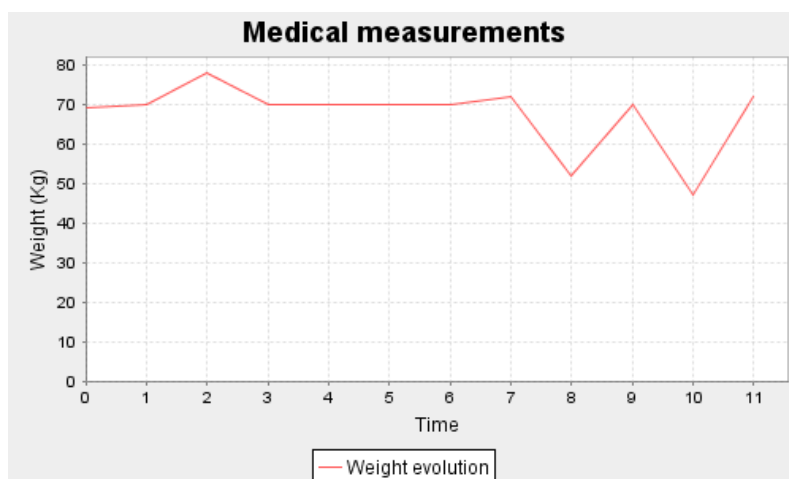


Figure 35 – Weight graphic.

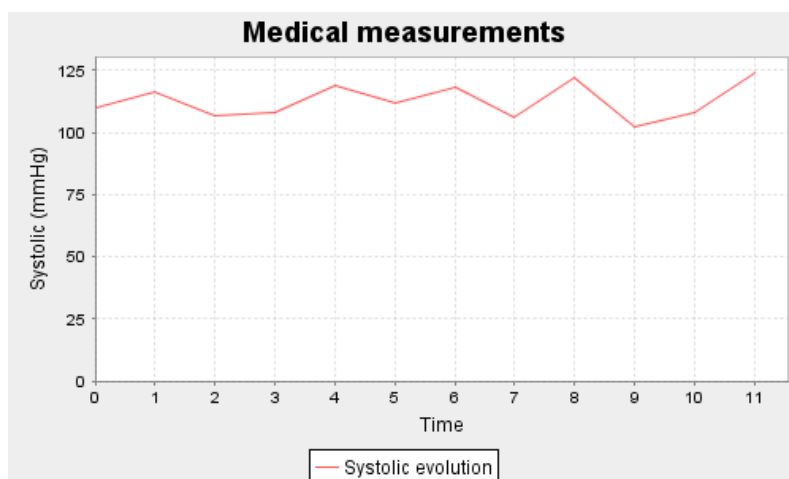


Figure 36 – Systolic graphic.

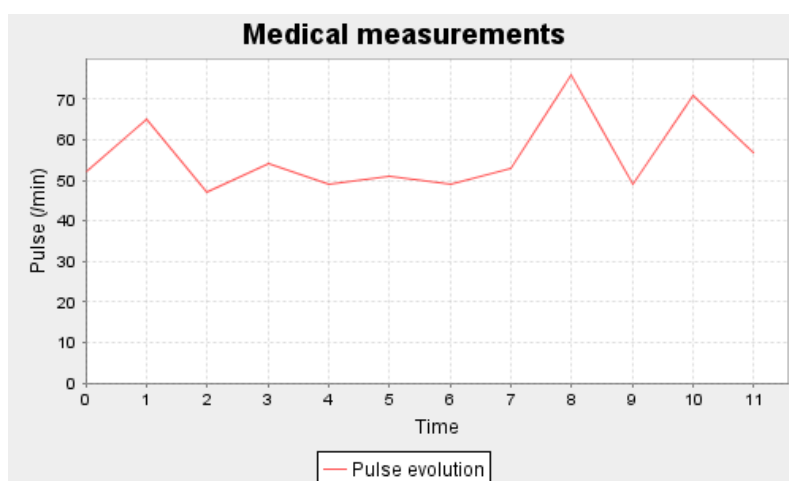


Figure 37 – Pulse graphic.

Chapter “Database and information access” described how the web pages are used to show the information in the patients’ EPR to medical staff. The following section explains how the dynamic web pages, the authentication, etc... have been implemented.

All the web pages used to access the patients’ EPR are codified in JSP format. JavaServer Pages is a Java technology that allows software developers to dynamically generate HTML, XML or other types of documents as a response to a Web client request.

JSP technology allows Java code and certain pre-defined actions to be embedded into static content. JSPs are compiled into Java Servlets by a JSP compiler. A JSP compiler may generate a servlet in Java code that is then compiled by the Java compiler, or it may generate byte code directly for the servlet. JSPs can also be interpreted on-the-fly reducing the time taken to reload changes.

For this application, medical staff authentication is static, meaning that login and password are defined in “redirect.jsp”. “index.jsp” is the default web page to access our system, requesting a login and a password to the person who tries to access. These data are sent through a form to “redirect.jsp”. The login data are checked, and if it is correct, the previous web page redirects to “IDRequest.jsp” otherwise it redirects to “index.jsp” again. Source code for “index.jsp” and “redirect.jsp” is as follows.

```
<html>
  <head>
    <title>Medical Measurements</title>
  </head>
  <body bgcolor="#66FFCC">
    <p>&nbsp;</p>
    <p>&nbsp;</p>
    <form method="POST" action="redirect.jsp">
      <p align="center">Login<input type="text" name="login" size="20"></p>
      <p align="center">Password<input type="password" name="pass" size="20"></p>
      <p align="center">
        <input type="submit" value="Send" name="B1">
        <input type="reset" value="Reset" name="B2">
      </p>
    </form>
  </body>
</html>
```

Source Code 13 – index.html web page

```
<html>
  <head>
    <title>Medical Measurements</title>
  </head>
  <body>
    <% if(request.getParameter("login").compareTo("Login")==0 &&
        request.getParameter("pass").compareTo("Password")==0)
      {
        String logeado = "true";
        session.setAttribute("logged", logeado);
      %>
      <script language="javascript">
        window.location="IDRequest.jsp";
      </script>
    <% }else{ %>
      <script language="javascript">
        window.location="index.jsp";
      </script>
    <% } %>
  </body>
</html>
```

Source Code 14 – redirect.html web page

Once medical staff is authenticated, the web server redirects to "IDRequest.jsp". Note that, this web page cannot be access without have been suceesfully logged in previously. If someone tries to open directly this web page from address bar, an error message will be shown. If the web page is accessed properly, it will ask for an ID patient. The source code for this page is as follows.

```
<html>
  <head>
    <title>Medical Measurements</title>
  </head>
  <body bgcolor="#66FFCC">
    <% String temporal = "" + session.getAttribute("logged");
        if(temporal.compareTo("true")==0) {
      %>
      <p>&nbsp;&nbsp;&nbsp;</p>
      <p>&nbsp;&nbsp;&nbsp;</p>
      <form method="POST" action="showReport.jsp">
        <p align="center">Patient ID<input type="text" name="dni" size="10"></p>
        <p align="center">
          <input type="submit" value="Accept" name="B1">
          <input type="reset" value="Cancel" name="B2">
        </p>
      </form>
    <% }else
      out.println("You are not allowed to acces this WebSite");
    %>
  </body>
</html>
```

Source Code 15 – IDRequest.html web page

```
<%
@ page contentType="text/html; charset=iso-8859-1" language="java" import="java.sql.*,
java.io.*, javax.xml.transform.*, javax.xml.transform.stream.*" errorPage=""
%>

<%
String temporal = "" + session.getAttribute("logged");
if(temporal.compareTo("true")==0){
    String path = application.getRealPath("reports/tmp.html");
    File lf_file = new File(path);
    FileOutputStream los_file = new FileOutputStream(lf_file);
    TransformerFactory tFactory = TransformerFactory.newInstance();
    Transformer transformer;
    try {
        String path01 = "./Medical/reports/cda.xsl";
        transformer = tFactory.newTransformer(new StreamSource(path01));
        transformer.transform(new StreamSource(application.getRealPath("reports/" +
            request.getParameter("dni") + ".xml")), new StreamResult(los_file));
    }
    <jsp:include page= "reports/tmp.html"/>
    <%
        }catch (Exception e) {
            out.println("Non patient found with this ID");
        }
        los_file.flush();
        los_file.close();
        lf_file.delete();
    }else
        out.println("You are not allowed to acces this WebSite");
    <%
%>
```

Source Code 16 – showReports.html web page

As it is shown in the previous code source the IDpatient information is sendt to “showReport.jsp”. This is the most complex web page in the server because it necessary to make a XSLT Transformation to show the reports. According to HL7 standards, reports in XML format have to be transformed with “cda.xsl”. It is a stylesheet which allows transforming XML reports to HTML code to be viewed for people in a readable format. Process XSLT Transformation:

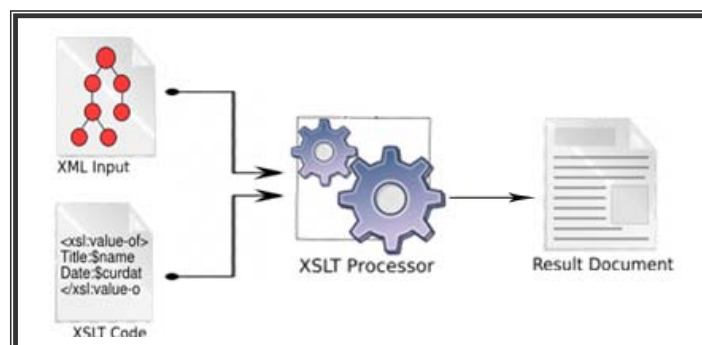


Figure 38 – XSLT Transformation.

6. REFERENCES

Barrett, D. J., R. E. Silverman, et al. (2001). "SSH, the secure shell the definitive guide." 1st. from <http://proquest.safaribooksonline.com/0596000111>.

González, J. F. and J. J. S. Ruiz (2007). Electrical Bioimpedance Measurement System for Limb Edema Monitoring. Borås, Högskolan i Borås.

Graham87. (2008). "Body weight." from http://en.wikipedia.org/wiki/Body_weight.

HL7_Staff. (2004). "What is HL7?" from <http://www.hl7.org/index.cfm>.

JForget. (2008). "Human weight." from http://en.wikipedia.org/wiki/Human_weight.

Kemp, B. and J. Olivan. (2003). "European data format 'plus' (EDF+), an EDF alike standard format for the exchange of physiological data." from <http://www.edfplus.info/specs/edfplus.html>.

MySQL_Staff. (2008). "Top Reasons to Use MySQL." from <http://www.mysql.com/why-mysql/topreasons.html>.

Ruben, D. (2006). "Blood pressure." from http://en.wikipedia.org/wiki/Blood_pressure.

Tlomak, L. (2008). "Data Encryption Standard (DES)." from http://en.wikipedia.org/wiki/Data_Encryption_Standard.

7. APPENDIX

Appendix I. UC-321PBT Precision Health Scale Specifications



Figure 39 – Bluetooth Blood Pressure Monitor.

Capacity	200 kg / 450 lb
Resolution	100g / 0.2 lb
Sensor	Load cell
Display	LCD, character height: 25mm
Power	4 type AA batteries (R6P)
Battery life	Approx. 2,000 measurements
Weight	2.7kg
Dimensions	320 x 314 x 35 mm (WxDxH)
Auto power (on / off)	45 seconds / 10 seconds
Operating temperature	10°~ 35°C (50°~ 95°F)
Communication	Bluetooth wireless technology (class 1, version 1.2)

Appendix II. UA-767PBT Blood Pressure Monitor Specifications



Figure 40 – Bluetooth Weight Scale.

Method	Oscillometric
Display Type	Digital, LCD16/10-mm character height, Pressure/pulse Displayed simultaneously
Measurement Range	20 – 280 mmHg (pressure) 40 – 200 pulse/minute (pulse)
Accuracy	± 3 mmHg or 2%, whichever is greater (pressure) ± 5% (pulse)
Pressurization	Automatic pressurization by micro pump
Depressurization	Constant speed exhaust valve
Deflation	Automatic exhaust by electromagnetic valve
Power supply	4 Type AA batteries (R6P)
Battery life	Approximately 300 measurements
Operating environment	+ 10°C ~ + 40°C (+ 50°F ~ 104°F) 30%RH to 85%RH
Storage environment	10°C ~ + 60°C (14°F ~ 140°F) 30%RH to 85%RH
Weight	Approximately 300g (0.66lb)
Dimensions	64(H) x 147(W) x 110(D) mm
Standard accessories	Soft carrying case, instruction manual & batteries
Options	AC adapter TB-182-C —230V, C plug, CE TB-182-BF—230V, BF plug , CE
Communication	Bluetooth wireless technology (Class1, version 1.2)